Artificial Intelligence 175 (2011) 2021–2060

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Algorithms and mechanisms for procuring services with uncertain durations using redundancy

S. Stein [a,*], E.H. Gerding [a], A.C. Rogers [a], K. Larson [b], N.R. Jennings [a]

[a] Intelligence, Agents, Multimedia Group, School of Electronics and Computer Science, University of Southampton, Southampton, United Kingdom
[b] Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

ARTICLE INFO

Article history:
Received 16 September 2010
Received in revised form 22 April 2011
Accepted 15 July 2011
Available online 23 July 2011

Keywords:
Mechanism design
Multi-agent systems
Service-oriented computing
Uncertainty
Redundancy

ABSTRACT

In emerging service-oriented systems, such as computational clouds or grids, software agents are able to automatically procure distributed services to complete computational tasks. However, service execution times are often highly uncertain and service providers may have incentives to lie strategically about this uncertainty to win more customers. In this paper, we argue that techniques from the field of artificial intelligence are instrumental to addressing these challenges.

To this end, we first propose a new decision-theoretic algorithm that allows a single service consumer agent to procure services for a computational task with a strict deadline. Crucially, this algorithm uses redundancy in a principled manner to mitigate uncertain execution times and maximise the consumer's expected utility. We present both an optimal variant that uses a novel branch-and-bound formulation, and a fast heuristic that achieves near-optimal performance. Using simulations, we demonstrate that our algorithms outperform approaches that do not employ redundancy by up to 130% in some settings.

Next, as the algorithms require private information about the providers' capabilities, we show how techniques from mechanism design can be used to incentivise truthfulness. As no existing work in this area deals with uncertain execution times and redundant invocations, we extend the state of the art by proposing a number of payment schemes for these settings. In a detailed analysis, we prove that our mechanisms fulfil a range of desirable economic properties, including incentive compatibility, and we discuss suboptimal variants that scale to realistic settings with hundreds of providers. We show experimentally that our mechanisms extract a high surplus and that even our suboptimal variants typically achieve a high efficiency (95% or more in a wide range of settings).

1. Introduction

Increasingly, participants in large distributed systems are able to discover and automatically procure the services of others. This allows service consumers to complete complex computational tasks on demand, but without the need to invest in and maintain expensive hardware. Already, such a service-oriented approach is gaining popularity in a large range of application areas, including grids, peer-to-peer systems, and cloud and utility computing [12,48,21].

Despite its benefits, flexible service procurement poses new challenges that have not been addressed satisfactorily by current research. In particular, as they are offered by external providers that are beyond the consumer's direct control,

* Corresponding author.
E-mail addresses: ss2@ecs.soton.ac.uk (S. Stein), eg@ecs.soton.ac.uk (E.H. Gerding), acr@ecs.soton.ac.uk (A.C. Rogers), klarson@cs.uwaterloo.ca (K. Larson), nrj@ecs.soton.ac.uk (N.R. Jennings).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2011.07.002

services may display significant uncertainty in their behaviour. Thus, the execution time of services can be highly uncertain, due to concurrent access by other consumers, hardware or network problems and the provider's scheduling policies. This is particularly problematic when services take a long time to complete, as is common for many computationally intensive tasks, and when consumers need to obtain their results by a certain deadline.

Furthermore, in large systems, many different providers may offer functionally equivalent services that are heterogeneous in their quality and costs. This requires consumers to make appropriate decisions about which services to procure, balancing the probability of success with the overall cost. In particular, instead of only procuring a single provider, the consumer may benefit by redundantly procuring multiple service providers that will attempt the same task. For example, when faced with a high-priority task but with a long deadline, a consumer may at first invoke an unreliable service at a low cost. However, as the time approaches the deadline and the task is still not completed, it may invoke another, more costly but also more reliable service to ensure that the task is completed in time. Alternatively, when a critical task has to be completed urgently, the consumer may be better off (both in terms of costs and probability of success) by selecting multiple cheap services immediately instead of a single premium service. This creates a challenge for the consumer, who has to make these decisions and wants to maximise its profit.

However, even when a consumer can make optimal decisions about which services to procure and whether to employ redundancy, it is faced with a second, highly related challenge. This is the fact that service providers are inherently self-interested agents and, thus, they may choose to misrepresent their capabilities if this promises to increase their profits. For instance, a provider may exaggerate its speed, in order to entice potential customers to procure its service, or it may inflate its costs to elicit higher payments. In these cases, consumers may end up procuring unsuitable services that are unable to complete the task in a timely or effective manner. Put differently, the consumer's decisions may be based on wrong information and therefore lead to suboptimal procurement strategies.

Clearly, both the problems of dealing with uncertain execution times and the providers' possible strategic behaviours are closely interrelated. More specifically, in order to address the first challenge satisfactorily, we also have to ensure that providers truthfully reveal their capabilities to the consumer. However, as we will see later, ensuring truthfulness, in turn, requires us to solve the service procurement problem optimally. For this reason, we address both of these intertwined challenges in this paper.

Now, there is an array of existing techniques that apply to our setting. In particular, decision theory and computational search techniques have been used to design agents that can take optimal actions in uncertain environments, while mechanism design has been employed successfully to incentivise truthfulness in multi-agent systems. Unfortunately, however, none of the existing approaches are readily applicable to the scenario we consider here (see Section 2 for details). For this reason, we combine and extend the current state of the art from multiple sub-fields of artificial intelligence and demonstrate how the resulting techniques can be applied to a realistic large-scale problem.

In more detail, to address the first problem of **uncertain execution times**, we make the following contributions in our work:

- We are the first to characterise the optimal solution to a generic service procurement problem where a service consumer can procure multiple service providers dynamically and redundantly over time, in order to complete a task by a given deadline. To find this efficiently, we combine analytical optimisation with computational search techniques. In more detail, we present a novel branch-and-bound algorithm that exploits specific characteristics of the procurement scenario and that we empirically show to reduce the search for the optimal solution, on average, by over 99.9%. As this algorithm relies on finding optimal procurement times from a continuous domain, we derive efficient closed-form solutions for both settings with (i) independent execution durations and with (ii) perfectly correlated durations.
- While our branch-and-bound algorithm quickly finds a solution in settings with dozens of providers, it does not scale to significantly larger systems. Hence, we also present a suboptimal heuristic algorithm to the service procurement problem. This combines some of our analytical results from the optimal algorithm with a greedy local search in a novel manner. As a result, it is capable of scaling to settings with hundreds or even thousands of providers.
- We evaluate both algorithms extensively using simulations. In doing this, we show empirically that they achieve an up to 130% improvement over techniques that do not use redundancy, that they also consistently outperform existing ad hoc techniques that are used in practice and finally that our heuristic solution achieves near-optimal performance. We also note that although our algorithms perform particularly well in environments where service durations are independently distributed, redundancy can still be beneficial in settings with perfect correlation, resulting in an average improvement of over 27% in certain scenarios.

As the first part of our work relies on having full information about the providers' capabilities, we also address the second interrelated problem of **strategic behaviour** with the following contributions:

- To apply our algorithms in settings with private information, we extend the state of the art in mechanism design and propose a number of novel incentive compatible mechanisms to deal with our service procurement problem. Unlike existing approaches, which have so far concentrated on services with a deterministic runtime and which allocate a task only to a single provider agent, our mechanisms can deal specifically with uncertain execution durations and multiple providers that execute the same task in parallel. We propose several mechanisms here, in order to effectively address a

wide range of settings with varying information and computational requirements. This is needed because a mechanism that implements the optimal procurement schedule is infeasible in settings with hundreds of potential providers and so we present a range of suboptimal alternatives for large-scale problems, some of which can use arbitrary heuristics to select an outcome if some information about the providers is known.[1]

In more detail, we first consider settings where only the cost is private information and then where both the cost and service duration distribution are private and need to be elicited. For the former case, we present several mechanisms that vary in their information requirements about the types of providers.

- A *uniform pricing* mechanism that is incentive compatible in dominant strategies and individually rational. Furthermore, these properties continue to hold when a suboptimal heuristic is used to find an allocation. However, to achieve a high efficiency (around 94%), this requires some prior knowledge about the distributions of provider types.
- Two types of *discriminatory pricing* mechanisms that have the same economic properties as the *uniform pricing* mechanism. In contrast, however, these require no specific knowledge about the provider types and still achieve a good efficiency (87–88% on average).

For settings where both costs and service duration distributions are private information, we present two further mechanisms:

- An *Execution-Contingent Vickrey–Clarke–Groves* (EC-VCG) mechanism that conditions payments on the earliest completion time of the task. This mechanism is ex-post incentive compatible and individually rational, but it requires an optimal solution to the service procurement problem.
- An *approximate* EC-VCG mechanism that retains the same economic properties as the EC-VCG, but that uses a suboptimal allocation function with a polynomial run-time (in the number of providers). We show experimentally that the loss of utility in using this approximation is typically small (less than 10% or better) and can be balanced explicitly with the computational effort required to find a solution.

We stress that our contributions are not limited to applications in service-oriented systems, but we believe this domain constitutes a compelling and timely motivation for our work. More generally, our techniques can be applied in many settings where a project or task of uncertain duration is outsourced. As an example of this, a critical product may need to be procured in a dynamic supply chain application — here, the consumer may be able to choose between different producers with uncertain production and delivery times, and the optimal strategy may include obtaining the product from multiple providers. In another application, a government may need to urgently find a vaccine for an epidemic and can contract different laboratories to work on this in parallel. Finally, another key application of our work is the emerging fields of crowdsourcing and human computation [47], where our work can be used to elicit uncertain completion times of a given task from a large pool of workers and even contract several workers in parallel when this is beneficial.

The remainder of this paper is structured as follows. In Section 2, we start by discussing related work, followed, in Section 3, by a formal summary of the problem we are addressing. In Section 4, we describe a generic approach for finding an optimal procurement strategy when the consumer has full information about the performance of services, and consider various models of uncertainty. Then, we extend this in Section 5 and present mechanisms to address settings with private information. In Section 6, we evaluate our strategy and mechanisms empirically and then conclude in Section 7.

2. Related work

The problem of allocating computational tasks to providers with uncertain execution is not new, and several researchers have addressed specific aspects of this problem in the past. This research has been carried out in a wide range of fields — some work has looked specifically at robustness in service-oriented systems, but other highly-relevant research has been conducted in the more general areas of scheduling and planning under uncertainty. Finally, the literature on mechanism design in multi-agent systems has considered settings where service provider agents need to be incentivised to reveal their capabilities truthfully, and has also considered settings with uncertainty. In the following, we discuss each of these strands of research in turn.

2.1. Robust service procurement

There is already a considerable body of work that suggests the use of redundancy to address uncertainty. This is based on techniques used in reliability engineering, where critical components are duplicated in order to increase the reliability of a system [54,8]. Now, while there are many analytical and heuristic tools in this area, they concentrate mainly on either minimising cost or failure probability, subject to constraints. We believe that this is insufficient and will concentrate instead on the expected utility of the service consumer, which implicitly balances the cost and failure probability of a task. Nevertheless, the work in reliability engineering has given rise to similar techniques in the context of services. In this vein, a critical task can be delegated to several unreliable service providers at once, which increases the overall success probability. Such parallel redundancy has been used successfully in a number of application examples, from deployed peer-to-peer systems [1], to highly dependable Web services [24] and multi-agent systems or software engineering in general [23,59].

[1] For a full overview of our novel mechanisms and their specific theoretical properties, see Table 2 in Section 5.1.

Instead of invoking service providers in parallel, other work has considered the serial invocation of services. Here, a particular provider may be contacted first, but if it fails or takes too long, the task is then delegated to another provider. A prolific example of this includes Google's MapReduce system, which uses this technique to execute large collections of data-processing tasks robustly on computational clusters [9], but this approach has also been suggested in the context of peer-to-peer systems [13]. Similarly, existing service frameworks often use pre-defined timeout values to determine when to switch to alternative service providers [38,10].

A major shortcoming with most work discussed so far, however, is its reliance on *ad hoc* techniques for choosing an appropriate number of redundant services and timeout values. A more principled approach is taken by work on restarting Web queries, which examines when such queries should be timed out and re-issued (possibly to a different provider) to ensure timely completion [6,32]. Similar work also exists in the domain of grid services [18]. However, such research typically assumes that only one query is active at any time and the costs of multiple queries are not explicitly balanced with the resulting benefit.

This shortcoming is addressed by Stein et al. [52], who use decision theory in this context to combine both parallel and serial redundancy to maximise the service consumer's expected utility. They show that this approach allows the consumer to successfully complete its tasks within strict deadlines even when dealing with providers that are failure-prone and display high duration uncertainty. In related work, they apply similar techniques to dynamic markets where service-level agreements can be negotiated in advance [51]. Our approach will be similar to theirs, but there are a number of key differences. First, they concentrate on large workflows of interdependent tasks, necessitating the use of suboptimal heuristics. In contrast, we will consider optimal algorithms for single tasks and present an analytical solution for a particular family of distributions. Second, their work assumes that probabilistic performance information about the service providers is available to the consumer. A key contribution of this paper is a number of mechanisms that incentivise providers to reveal this information truthfully to the consumer.

2.2. Stochastic scheduling and planning

Research on stochastic scheduling also bears some similarities to our work. Here, the goal is to find optimal schedules or policies for completing tasks with stochastic durations or random failures on one or more processors [42,3]. Most of this research does not consider redundant execution of tasks on multiple processors, and even when it is considered, the model and objective functions are typically very different from our problem [37,44]. In particular, such work usually aims to maximise the total number of failures that can be tolerated by a scheduling algorithm. Often there are restrictions on the level of redundancy (typically it is limited to one primary and one backup processor) and it is assumed that the scheduling mechanism has complete control and free access to all processors. None of these considerations and assumptions apply to our service procurement scenario.

Somewhat closer to our work is the research on algorithm portfolios [22,19,27], where multiple instances of stochastic search algorithms are combined to produce faster and more dependable results. This is achieved by running the instances on independent processors or by interleaving and restarting them on a single processor. However, there are several important differences to our work. First, it is typically assumed that the use of processing resources is inherently free and the main objective is to minimise the run-time (or variance) over these resources. This is unrealistic in our settings, where resources are offered by self-interested agents. One exception to this is the work by Finkelstein et al. [11], which is closer to ours and explicitly considers the cost of running resources. However, like most other work on algorithm portfolios, they also assume that processes can be stopped and restarted at will. Such fine-grained control is often impractical in service-oriented systems, where providers may be unwilling to surrender control over their own scheduling policies. Furthermore, they consider only two parallel processors and even in this case, their problem is difficult to solve optimally in practice.

Another body of work that is relevant to the problem addressed in this paper is concerned with planning under uncertainty, especially when this explicitly considers continuous-time domains [31,33]. This research uses variations of Markov decision problems (MDPs), but such formalisms have a number of drawbacks. First, they are very general and require a number of adaptations to handle our scenario, including concurrently running and interruptible actions (i.e., to represent parallel and serial redundancy), which quickly leads to an infeasibly large state space as the number of providers increases. This is further exacerbated by the need to discretise action duration functions, which again increases the state space considerably and also results in suboptimal policies.

2.3. Mechanism design

So far, the work discussed in this section has concentrated on designing agent strategies to deal with uncertainty. When this uncertainty was explicitly modelled, it was typically assumed that some probabilistic information about the providers' behaviour was available from past interactions or through an appropriate trust and reputation system [45,53,20]. However, this might be unrealistic in settings where interactions are very costly, where no appropriate reputation system is in place, or where new providers routinely enter the system.

For such cases, some existing research has concentrated on designing appropriate mechanisms that incentivise providers to reveal private information about their cost and probabilistic performance estimates to the consumer. In particular, Porter et al. [43] suggest a mechanism that incentivises providers to report a truthful estimate of their success probability for

a given task. This is achieved by conditioning the payment to providers on whether they successfully complete the task. Ramchurn et al. [46] extend this by considering scenarios where providers also report on their perceived reliability of other providers. While they consider potential failures, these approaches do not use redundancy to increase the consumer's success probability and they also do not consider uncertain service durations.

Witwicki and Durfee [58] do investigate uncertain durations and propose a negotiation mechanism by which providers agree to provide a service by a given time with a certain probability. But their approach assumes that providers are cooperative (i.e., report their success probabilities truthfully), which is unrealistic in the open distributed systems we consider, and they also do not employ redundancy. Gerding et al. [15] present mechanisms for incentivising non-cooperative providers to invest costly resources in order to increase the probability of success of a task, and they also use redundancy to increase the utility of the consumer. Similarly, Babaioff et al. [2] consider a general principal-agent setting where multiple service providers need to be incentivised to exert some effort to complete a common task, but their individual actions cannot be verified and can have probabilistic outcomes. As a special case, their model covers the setting where multiple providers work on the same task redundantly. However, they assume that success probabilities and costs are publicly known. Furthermore, as in Gerding et al. [15], a service invocation either fails or succeeds, and they do not consider *time-critical* tasks. Therefore, the issue of *when* to invoke a certain provider is not investigated. A slightly different problem is investigated by Papakonstantinou et al. [40], who use scoring rules to incentivise agents to provide costly estimates of some probabilistic parameter. However, their setting is different in that the time to find a solution is not considered and in that multiple estimates can be merged to improve the overall quality.

Several mechanisms have been proposed specifically for the domain of computational services [5]. For example, Garg et al. [14] describe how a continuous double auction can be used to allocate resources in a computational grid, while Narahari et al. [35] discuss incentive-compatible mechanisms for a similar application. These approaches do not directly apply to the problem we consider, because they do not deal with uncertain execution times. Now, a mechanism that is closer to our work is used by Stein et al. [49]. Here, the authors investigate how service providers can be incentivised to reveal costs and processor speeds truthfully, but their work only considers uncertainty in the computational requirements of a task and not in the execution times of each provider. Their model is also fundamentally different to ours in that they allow tasks to be suspended and migrated from one provider to the next.

In our work, we address several of the above shortcomings. First, we design an optimal agent strategy that explicitly balances the benefit and associated cost of redundancy using a decision-theoretic approach, and we allow several providers to be invoked dynamically over time. Then, we develop a number of mechanisms that incentivise providers to reveal their private performance information to the consumer, so that a good procurement strategy with redundancy can be found.

3. Problem specification

In this section, we begin by introducing the problem in formal terms, and then present the utility functions that describe the preferences of the agents in the system and the social welfare that represents the utility of the system as a whole. We then consider various scenarios in which execution uncertainty occurs in practice, and generate a number of more specific models based on different assumptions about the nature of this uncertainty.

3.1. General setting

We consider a single service consumer A, who would like to complete a task T. The consumer derives a value $V \in \mathbb{R}^+$ if the task is successfully completed within a given deadline $D \in \mathbb{R}^+$, and 0 otherwise. The problem faced by the agent is that it is unable to execute the task on its own, and must procure the services of a third party. We assume that there are m service providers, given by the set $M = \{1, \ldots, m\}$, which can complete the task on the consumer's behalf. The consumer can invoke a provider $i \in M$ at any time in the interval $[0, D]$. In particular, the consumer may have multiple services running concurrently for the same task. In this case, the value V is obtained if at least one of the invoked services completes within the required time (and no additional value is obtained if multiple services complete the task). We assume that, once invoked, the provider remains committed to the task until it is completed (possibly beyond the deadline). Thus, a service cannot be interrupted.

As service completion times are generally uncertain, we let X_i be a random variable describing the execution time of provider i, where we assume that $\text{Prob}(X_i \leqslant 0) = 0$. This is the time from invocation to completion and includes any time needed for pre- and post-processing, queueing and data transfers. The random variables X_i for $i \in M$ are distributed according to the cumulative distribution functions $F_i(t)$, where $F_i(t) = \text{Prob}(X_i \leqslant t)$ is the probability that the task is successfully completed at most t time units after invocation. Furthermore, we let $f_i(t) = \mathrm{d}F_i(t)/\mathrm{d}t$ be the corresponding probability density function. In the following, we also refer to F_i as provider i's *duration function*. We assume, in general, that neither the consumer nor the providers can actively influence the running time. However, we do not necessarily assume that the running times of different providers are independently distributed. In Section 3.3, we will elaborate on how this uncertainty may arise in practice and how this affects the modelling assumptions.

While executing, provider i incurs a cost c_i, where this cost may represent both the running costs of its computational resources and opportunity costs from not being able to use these resources for other tasks. In the case that the cost is uncertain, because opportunities may or may not arise and because it typically depends on the execution time, c_i can

also be interpreted as the expected cost. For simplicity, however, we refer to c_i as a deterministic value. To compensate a provider for this cost, each provider $i \in M$ receives a payment, which is given by transfer functions τ_i. In the case that the provider costs c_i are publicly known, these transfers are simply equal to the costs when a provider has been invoked, and zero otherwise, regardless of whether the task succeeded in time.[2] While such a setting, where providers are only paid their costs, is unrealistic in many settings, we will investigate it in detail in Section 4. This allows us to compute the optimal procurement schedule with known costs and it will form the basis for private information settings. Formally, when c_i is known:

$$\tau_i = \begin{cases} c_i & \text{if provider } i \text{ is invoked} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

On the other hand, if the costs are *private*, the transfers are determined by a *procurement mechanism*, as discussed in Section 5, and they are typically not equal to the provider's costs. In this section we make no particular assumption regarding the transfer functions, only that these are calculated ex-post and can depend on all available information, including whether or not the task succeeded within the deadline. Note that transfers can also be negative, in which case the provider incurs a *penalty*. Finally, we assume that all participants are expected utility maximisers.

At this point, it should be noted that the described model has a number of possible extensions. First, in many applications, a consumer will often face workflows of several interdependent tasks rather than just a single task. Furthermore, we make the assumption that service costs stay constant throughout the active period of a task (i.e., between when it first becomes available and its deadline). However, in practice, varying demand and capacity constraints may lead to more dynamic costs[3] and there may even be uncertainty about future costs. Similarly, we assume a fixed deadline and that a successful service result can be easily verified by the consumer, but potential future extensions of our work could examine soft or uncertain deadlines and settings where there can be uncertainty about whether a task was completed successfully or not.

3.2. Procurement strategy and agent utility functions

Given the above setting, we are interested in finding a procurement strategy ρ, which specifies a plan that determines which providers should be invoked and when. We denote $\mathcal{P}$ to be the set of all valid procurement strategies, and compactly represent an element of this set as follows:

Definition 1 *(Procurement strategy).* A procurement strategy is a vector $\rho = \langle (s_1, t_1), \ldots, (s_n, t_n) \rangle \in \mathcal{P}$ with $n \leqslant m$, where each element represents the invocation time $t_i \in [0, D]$ of a provider $s_i \in M$. Importantly, a provider s_i is only invoked at time t_i (and incurs cost c_{s_i}) if no provider has so far completed the task. Without loss of generality, we assume that $t_i \leqslant t_{i+1}$ (i.e., elements of the vector are ordered by their invocation time), and $s_i \neq s_j$ if $i \neq j$. We use $\rho = \emptyset$ to denote the case where no provider is invoked.

For example, assume there are four providers, $M = \{1, 2, 3, 4\}$, and $\rho = \langle (2, 0),\ (3, 0),\ (1, 2.5) \rangle$. Here, providers 2 and 3 are invoked immediately. Then, if the task has not been completed by $t = 2.5$, provider 1 is also invoked, causing the three providers to run concurrently. Provider 4 is never invoked.

Given a strategy ρ and the random variables describing the duration distributions, we would like to derive the probability that task T is completed by a certain time. This is given by:

Definition 2 *(Completion probability).* Denote by $X_\rho = \min_{i \in \{1,\ldots,n\}}(t_i + X_{s_i})$ the random variable describing the completion time of the task. Then, the probability that the task T is completed by a certain time t is given by (recalling that $\text{Prob}(X_{s_i} \leqslant 0) = 0$):

$$\text{Prob}(X_\rho \leqslant t) = \text{Prob}\left(\bigcup_{i=1}^{n} X_{s_i} \leqslant t - t_i \right) \tag{2}$$

We are now ready to specify the utility functions of the consumer and the providers. In what follows, we specify the utility after execution of the procurement schedule, as well as the *expected utility* for a given procurement strategy *prior to its execution*.

Definition 3 *(Consumer's utility).* The consumer's utility is:

$$u_A(\rho) = \begin{cases} V - \sum_{i \in M} \tau_i & \text{if } X_\rho \leqslant D \\ -\sum_{i \in M} \tau_i & \text{if } X_\rho > D \end{cases} \tag{3}$$

[2] Note that we have assumed that a task is always completed by a provider, even if this is after the deadline. To prevent a service provider from not investing any resources and thus not incurring costs, the payment can be made after completion.

[3] This has partly been addressed in related work [49].

and the consumer's *expected* utility prior to execution is:

$$\bar{u}_A(\rho) = V \cdot \text{Prob}(X_\rho \leqslant D) - \sum_{i \in M} \bar{\tau}_i(\rho) \tag{4}$$

Here, $\bar{\tau}_i(\rho)$ is the *expected transfer* to providers i given schedule ρ prior to its execution. The transfers depend on the details of the procurement mechanism that is used, but for the setting where costs are publicly known, the expected transfer to provider i is simply its cost multiplied by the probability that it is invoked (which is equal to the probability that no other provider has completed the task by the time that provider i is scheduled to be invoked). In more detail, let $t_\rho(i)$ indicate the invocation time of provider i in the procurement strategy ρ, where we define $t_\rho(i) = \infty$ if provider i is not part of the schedule. Given this, $\bar{\tau}_i$ for the *full information setting* is calculated by[4]:

$$\bar{\tau}_i(\rho) = c_i \cdot \left[1 - \text{Prob}\big(X_\rho \leqslant t_\rho(i)\big)\right] \tag{5}$$

Furthermore, a service provider's utility is defined by:

Definition 4 *(Service provider's utility).* Let $\mathcal{I}_\rho = \{s_i\colon\ t_i \leqslant X_\rho\}$ be the set of providers that are invoked for the task. The utility of service provider i is:

$$u_i(\rho) = \begin{cases} \tau_i - c_i & \text{if } i \in \mathcal{I}_\rho \\ \tau_i & \text{otherwise} \end{cases} \tag{6}$$

and provider i's *expected* utility prior to execution of strategy ρ is:

$$\bar{u}_i(\rho) = \bar{\tau}_i(\rho) - c_i \cdot \left[1 - \text{Prob}\big(X_\rho \leqslant t_\rho(i)\big)\right] \tag{7}$$

In general, we are interested in choosing a strategy which maximises the *social welfare*, which is the sum of all utilities that agents derive in the system. This is a natural metric for how efficiently the agents work together in achieving the task, as it explicitly balances the benefit of successful task completion with the overall costs that are incurred. Now, since the actual completion time is unknown until execution, we need to consider the *expected* social welfare when selecting a strategy ρ. As such, this explicitly models the uncertainty in service execution times, which is a key consideration in our work, as described in Section 1. In what follows, we provide the equations for both the welfare and the expected welfare in turn. Note that these equations no longer contain the transfers, because these simply re-distribute utility between the agents.

Definition 5 *(Social welfare).* The social welfare is given by:

$$w(\rho) = u_A(\rho) + \sum_{i \in M} u_i(\rho) = \begin{cases} V - \sum_{i \in \mathcal{I}(\rho)} c_i & \text{if } X_\rho \leqslant D \\ -\sum_{i \in \mathcal{I}(\rho)} c_i & \text{if } X_\rho > D \end{cases} \tag{8}$$

The *expected* social welfare prior to execution is given by:

$$\overline{w}(\rho) = \bar{u}_A(\rho) + \sum_{i \in M} \bar{u}_i(\rho) = V \cdot \text{Prob}(X_\rho \leqslant D) - \sum_{i=1}^{n} c_{s_i} \cdot \big(1 - \text{Prob}(X_\rho \leqslant t_i)\big) \tag{9}$$

So far, we have not detailed how to calculate the probabilities in the above equations, in order to avoid making any assumptions about the duration probabilities (e.g., whether service durations of different providers are independent or whether they are correlated). We explore this issue further in the following section, where we instantiate Eq. (2) (completion probability) for particular environments. This will then allow us to analytically derive the optimal invocation times of providers under certain conditions in Section 4.

3.3. Models of uncertainty

As a fundamental part of our model, we assume that the execution time of a service is uncertain, but so far we have not been explicit about the nature of this uncertainty. In the following, we explore this issue in more detail and identify three possible levels of correlation that occur in practice: environments with no correlation between the probability distributions of the execution time, with perfect correlation, and with imperfect correlation. Furthermore, we describe a number of example scenarios where these different modelling assumptions are likely to apply in practice (or at least provide a good approximation) and for the first two settings we re-write Eqs. (2) and (9). This will then help us derive the optimal procurement strategy for these settings in Section 4.

[4] Note that in the special case where $\rho = \emptyset$, the transfers are always 0, i.e., $\bar{\tau}_i(\emptyset) = 0$.

3.3.1. Independent distributions

Most of the research on task allocation with execution uncertainty assumes that the execution durations of different providers are independently distributed. This is a reasonable assumption in a setting where, for example, the computational requirements for the task are known (in terms of the number of computational cycles), but there is uncertainty about the *load* on the service provider's resources at the time of execution. This may occur when the task is submitted to a queue on a computational cluster or mainframe, and there is uncertainty about the completion time of other tasks. Alternatively, the resource may be shared and concurrently running tasks, submitted by other users, may affect the duration of the consumer's task. As a result of this local, provider-specific uncertainty, it is reasonable to assume that service durations are independently distributed, i.e., that there is no correlation between the execution times of different providers.[5]

For a setting where independence can be assumed, Eq. (2) can be re-written as follows:

$$\begin{aligned}\text{Prob}(X_\rho \leqslant t) &= \text{Prob}\left(\bigcup_{i=1}^{n} X_{s_i} \leqslant t - t_i\right) = 1 - \text{Prob}\left(\bigcap_{i=1}^{n} X_{s_i} > t - t_i\right) \\ &= 1 - \prod_{i=1}^{n}\left(1 - F_{s_i}(t - t_i)\right) \end{aligned} \tag{10}$$

As a result, and given that $F_i(x) = 0$ for $x \leqslant 0$, Eq. (9) for the expected social welfare becomes:

$$\overline{w}(\rho) = V\left(1 - \prod_{i=1}^{n}\left(1 - F_{s_i}(D - t_i)\right)\right) - \sum_{i=1}^{n} c_{s_i} \prod_{j=1}^{i-1}\left(1 - F_{s_j}(t_i - t_j)\right) \tag{11}$$

3.3.2. Perfect correlation

In contrast to the provider-specific uncertainty, in other settings, the uncertainty may be associated with the task itself. That is, the task *difficulty* (e.g., in terms of the number of computational cycles required to solve it) is unknown a priori and only given by a probability distribution. For example, it is generally unknown how difficult specific instances of NP-hard optimisation problems are to solve, and techniques to estimate the *empirical hardness* [30] can be used to obtain a probability distribution of this difficulty. Assuming that resources are immediately available and not influenced by concurrent tasks, the duration of a service can then be modelled by a deterministic function of this difficulty. Generally, however, as providers may use different hardware, the processing speeds and costs between providers can vary considerably for a particular difficulty. Given this, in such a setting, the execution times of different providers are perfectly correlated.

We will now use an example scenario to describe this setting more formally. Let Y denote a continuous random variable describing the task difficulty, which is distributed according to the cumulative function $G(y)$ and corresponding density function $g(y)$ with support $[0, \infty]$. The task difficulty is defined by some commonly agreed metric (the details of which are not important for the purpose of our model). Each provider i has a *quality of service* function $q_i : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, where $q_i(t)$ denotes the *maximum* difficulty that provider i can solve within t time units. We assume that q_i is increasing and differentiable, and $q_i(0) = 0$. Furthermore, we define $F_i(t) = G(q_i(t))$ as the probability that provider i completes the task within t time units (which corresponds exactly to the duration function introduced at the beginning of Section 3).

Given this formalisation, we can re-write Eqs. (2) and (9) for this setting (as we show later, in Section 4, these equations can be simplified further for the optimal solution). In doing so, as a result of the perfect correlation, note that the union operator can be replaced by a maximisation, because only the provider that solves the highest difficulty in its allocated time (as given by $q_i(t)$) influences the overall success probability:

$$\begin{aligned}\text{Prob}(X_\rho \leqslant t) &= \text{Prob}\left(\bigcup_{i=1}^{n} X_{s_i} \leqslant t - t_i\right) = \text{Prob}\Big(Y \leqslant \max_{1 \leqslant i \leqslant n} q_{s_i}(t - t_i)\Big) \\ &= G\Big(\max_{1 \leqslant i \leqslant n} q_{s_i}(t - t_i)\Big) = \max_{1 \leqslant i \leqslant n} F_{s_i}(t - t_i) \end{aligned} \tag{12}$$

And Eq. (9) becomes:

$$\overline{w}(\rho) = V \cdot \max_{1 \leqslant i \leqslant n} F_{s_i}(D - t_i) - c_{s_1} - \sum_{i=2}^{n} c_{s_i} \cdot \Big[1 - \max_{1 \leqslant j \leqslant i-1} F_{s_j}(t_i - t_j)\Big] \tag{13}$$

[5] Even in such a setting, however, this assumption may not be valid. It could be the case, for example, that all the providers are highly loaded in busy periods, and the opposite holds in quiet periods. In this case, *conditional* independence may still be obtained, based, for example, on the time of day.

3.3.3. Imperfect correlation

While the previous two settings considered either perfect correlation between service durations or none at all, many scenarios will contain correlation to a limited extent. For example, the computational difficulty of a problem may still be uncertain, as described before, but different providers may use different algorithms to solve this problem. This could lead to imperfectly correlated execution times, as a generally hard problem will take longer to solve for all providers, but some providers may be faster on certain problems, but slower on others. Furthermore, the algorithm itself may be nondeterministic or the factors mentioned in Section 3.3.1 could contribute to the duration uncertainty of a particular provider.

Although such settings with imperfect correlation are very common, they typically require a much more domain-specific model that describes the nature of the correlation between service durations. As a result, we do not analyse imperfectly correlated settings in this paper, and instead concentrate on the former two settings. However, many of our results can be extended to settings with imperfect correlation. In particular, the mechanisms presented in Section 5, which incentivise the providers to truthfully report their costs and duration functions, also apply to the imperfect correlation case. However, this does require that the problem can be solved *optimally within the range of allowable outputs* (this is described in more detail in Section 5). Since imperfect correlation settings are unlikely to admit an analytical solution, the time space may need to be discretised.

This concludes our discussion of the problem that we address in the paper, the utility functions that motivate agents in the systems we consider, and the various types of execution uncertainty that may occur in practice. In the next section, we describe how an optimal procurement strategy can be found by the service consumer.

4. Optimal service procurement

Finding an optimal procurement strategy, i.e., deciding which providers to invoke at what times, is a fundamental part of this paper. First, it is of interest in current service markets, where providers typically offer their resources at fixed prices that are publicly listed (i.e., where transfers are not determined by a mechanism, but where they are set by providers, depending on the balance of supply and demand). Second, it will allow us to design *efficient* mechanisms in Section 5. In this context, we are mostly interested in maximising the expected social welfare, because this represents how well the available providers are used to complete the task and explicitly balances the combined costs of the providers with the value of successful execution.

Throughout this section, we will assume that both the providers' costs c_i and duration functions F_i are public information and that the consumer simply compensates providers for their incurred costs when they are invoked (i.e., $\tau_i = c_i$ if provider i is invoked and $\tau_i = 0$ otherwise). In Section 5, we will describe settings where this assumption does not hold. For now, it means that the social welfare and the consumer's utility are equal, i.e., $\overline{w}(\rho) = \bar{u}_A(\rho)$, and we are seeking an optimal strategy $\rho^* = \text{argmax}_{\rho \in \mathcal{P}} \overline{w}(\rho)$ that maximises these.

Before we show how ρ^* can be found, we first outline an example scenario. This highlights the potential benefit of redundancy by showing that appropriate procurement of multiple service providers can lead to a significant improvement in utility compared to relying only on a single service provider.

4.1. Motivating example

In this example, a graphic designer needs to render a high-resolution image for an advertising campaign. As she needs to present her work to senior managers at a board meeting later that day, she has a strict deadline of 60 minutes ($D = 60$) and values the task at \$100 ($V = 100$).

Now, to complete the rendering task, she has several computational resources at her disposal. As her first option, she can submit the task to one of several desktop PCs within her company, which are set up to use idle processing cycles to run background tasks. These are cheap (her department is billed \$0.60 each time a task is submitted, i.e., $c_{\text{PC}} = 0.6$), but the completion time is highly uncertain, as the desktop may be in use by its regular owner and other concurrent tasks. Here, we assume that this uncertainty is described by an exponential distribution with a mean duration of 2 hours, i.e., $F_{\text{PC}}(t) = 1 - \mathrm{e}^{-\frac{t}{120}}$.

As her second option, she can outsource the task to a powerful mainframe computer, which is maintained by an external company. This is significantly faster, due to its powerful hardware and its generally lower congestion. Hence, we here assume the distribution $F_{\text{MF}}(t) = 1 - \mathrm{e}^{-\frac{2t}{3}}$, with a mean duration of only 1.5 minutes. However, as the mainframe is expensive to run, the cost is higher, with the provider charging \$60 for an invocation ($c_{\text{MF}} = 60$).

As is common in related work, the designer may now choose the service provider that promises the highest expected utility. Using one of the desktop PCs would result in $\overline{w} = V \cdot F_{\text{PC}}(D) - c_{\text{PC}} = 38.75$, while submitting the task to the external mainframe would yield a slightly higher utility of $\overline{w} = V \cdot F_{\text{MF}}(D) - c_{\text{MF}} = 40$. Hence, she would be best off choosing the latter option, which will virtually guarantee successful execution by the deadline, but also incur a large cost.

However, instead of choosing one provider, we now demonstrate how she could exploit redundancy to complete the task. Here, we assume that there are three identical desktop PCs within her company ($i \in \{1, 2, 3\}$), as well as the external mainframe ($i = 4$). For now, let us assume that durations are independently distributed, as described in Section 3.3.1. Given this, she can submit the job to several PCs and then later to the mainframe, to ensure that the task is completed in time. In

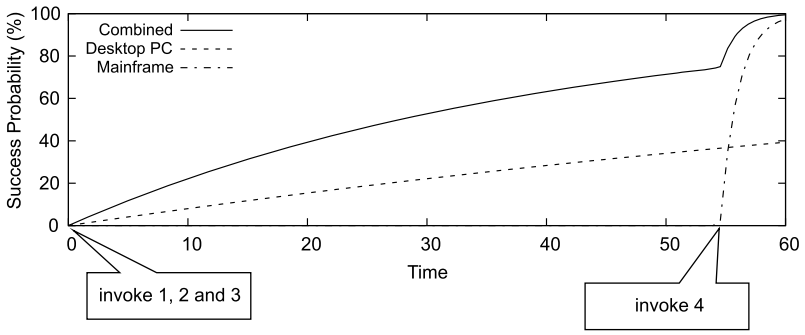

Fig. 1. Success probability over time for ρ^* (combined) and its constituent services (desktop PC and mainframe) when durations are uncorrelated.

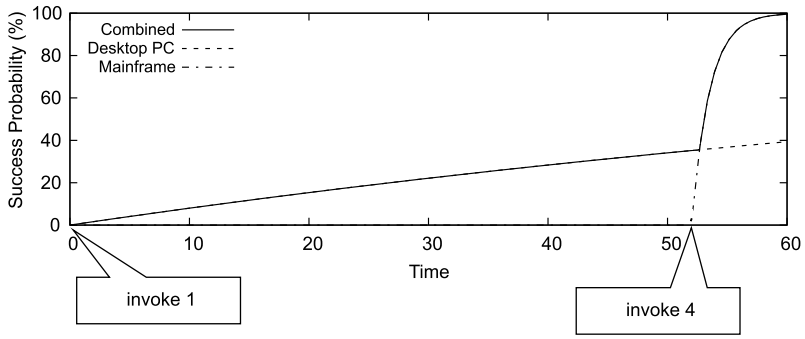

Fig. 2. Success probability over time for ρ^* (combined) and its constituent services (desktop PC and mainframe) when durations are perfectly correlated.

fact, in this setting, the optimal strategy for her is $\rho^* = \langle (1,0), (2,0), (3,0), (4,54.51) \rangle$, which results in an expected utility of $\overline{w} = V \cdot (1 - (1 - F_{\text{PC}}(D))^3 \cdot (1 - F_{\text{MF}}(D - 54.51))) - 3c_{\text{PC}} - c_{\text{MF}} \cdot (1 - F_{\text{PC}}(54.51))^3 = 82.27$. This strategy is shown in Fig. 1, which plots the success probability over time for ρ^*, as well as for the constituent services that are invoked as part of ρ^*. Overall, ρ^* results in more than a 100% improvement over the best single provider strategy.

Next, we examine the setting where service durations are perfectly correlated, as described in Section 3.3.2. To this end, we use the same parameters as given above ($D = 60$ and $V = 100$), but now assume that all uncertainty lies in the difficulty of the rendering task. More specifically, the designer now knows that all potential machines are idle, but she is unsure about the number of processing cycles needed to complete the task. Thus, we assume that the difficulty of the task is distributed according to $G(y) = 1 - \mathrm{e}^{-y}$. Furthermore, we assume that the same local PCs and an external mainframe are available and that their service durations are given, respectively, by the functions $\text{PC}(y) = 120y$ (where y is the task difficulty) and $q_{\text{MF}}(t) = \frac{2}{3}t$. This results in exactly the same distribution functions as above, but the durations are now perfectly correlated.

If only one provider is invoked in this setting, the best expected utility is again gained by invoking only the expensive mainframe, resulting in $\overline{w} = 40$. When employing redundancy, we first note that, in this setting, the designer cannot gain anything by invoking more than one of the cheap desktop PCs, as their durations are always identical. However, she can still benefit from first invoking a cheap PC and then later a mainframe. This is because the mainframe may never be needed, but if the task is still not completed just before the deadline, it can be invoked to increase the probability of success. More specifically, the optimal strategy in this case is $\rho^* = \langle (1,0), (4,52.01) \rangle$, which results in an expected utility of $\overline{w} = 60.02$ (see Fig. 2). While this is not as large as in the uncorrelated case, it still constitutes a 50% improvement over the single provider approach.

These examples highlight the potential benefit of using redundancy for task procurement scenarios, both in the cases where durations are independent and when they are perfectly correlated. In the following, we discuss how an optimal procurement strategy can be found in both cases. More specifically, we split the problem into two distinct parts and start in Section 4.2 by looking at the problem of computing the optimal invocation times, given that we already know which providers should be invoked and in what order. In Section 4.3, we then describe how this ordering of providers can be found.

4.2. Optimal invocation times

We are now interested in finding an optimal procurement strategy ρ^*, but we note here that this is a computationally hard problem, due to its combinatorial nature and the nonlinear objective function. We also make the observation that it a specific case of the restless bandit problem [57]. The restless bandit problem is an extension of the multi-armed bandit

problem where there are n arms and at any time t, some number M of those arms can be chosen to be activated. When an arm in state i is activated, a cost c_i is incurred and it transitions to state j with probability p_{ij}. If an arm is not activated, then no cost is incurred and it transitions to state j with probability q_{ij}. The goal is to find a policy or schedule for activating the arms so as to maximise average rewards. In Appendix A, we describe how the optimal procurement problem can be modelled as a restless bandit problem. Classic multi-armed bandit problems are a special case of the restless bandit problem where $q_{ii} = 1$ and $q_{ij} = 0$ for all $i \neq j$, but while the multi-armed bandit problem has an optimal solution by using the Gittins priority-index rule [17] for which there are polynomial time algorithms, this is not the case for the restless bandit problem. In particular, it has been shown that even in the case where only one arm is activated at a time ($M = 1$) *and* all transitions are deterministic, the problem of computing an approximately optimal schedule is PSPACE-hard [39].[6] Since the optimal procurement problem is an instance of a restless bandit problem, where $M = 1$ but allowing for probabilistic transitions, we can make the following observation:

Observation 1. The optimal procurement problem can be modelled as a restless bandit problem and it is PSPACE-hard to compute an approximately optimal schedule.

To solve the problem, we initially assume that the optimal subset of providers and their ordering is given. That is, we are given an ordered set of providers $\rho_s^* = \langle s_1, \ldots, s_n \rangle$ where s_i is invoked before s_{i+1}. To compute the optimal procurement schedule, we must determine $\rho_t^* = \langle t_1, \ldots, t_n \rangle$, where t_i is the invocation time of s_i. To this end, we compute the gradient of the expected welfare, $\nabla \overline{w}(\rho_t^*)$, and find its root, i.e., $\nabla \overline{w}(\rho_t^*) = \mathbf{0}$ (if it exists). This results in a system of n simultaneous equations, with one equation for each t_i, with constraints, $\forall i\colon\ 0 \leqslant t_i \leqslant D$, and $\forall i, j : i \leqslant j \leftrightarrow t_i \leqslant t_j$. Solving these equations, checking the appropriate second order conditions and identifying the global maximum depend on the family of duration distributions and can be done either analytically or numerically using standard optimisation software (as available, e.g., in Matlab, Mathematica or Maple).

In what follows, we will make a number of assumptions about the duration distributions, in order to derive closed-form analytical solutions. Specifically:

- We will focus on the exponential distribution, as this is commonly used for modelling uncertain service durations [55].[7]
- As discussed in Section 3.3.3, we will concentrate on the two scenarios of independent and perfectly correlated durations. We will cover these separately, as the objective functions are fundamentally different.

We stress that these two assumptions do not limit the generality of our approach and the mechanisms outlined in Section 5. They simply allow us to derive simple closed-form solutions for the optimal invocation times in this section. To this end, in the following we start by looking at the setting with independent durations.

4.2.1. Independent durations

We now derive analytical expressions for the invocation times ρ_t^*, given ρ_s^* and given that the duration distributions of providers $i \in M$ are described by $F_i(t) = 1 - \mathrm{e}^{-\lambda_i t}$, where $\lambda_i > 0$ is a rate parameter. Re-writing Eq. (11) with these distributions, and computing the gradient now yield:

$$\frac{\partial \overline{w}(\rho)}{\partial t_i} = -V \cdot \lambda_{s_i} \prod_{j=1}^{n} \mathrm{e}^{-\lambda_{s_j}(D-t_j)} + c_{s_i} \sum_{j=1}^{i-1} \lambda_{s_i} \prod_{k=1}^{i} \mathrm{e}^{-\lambda_{s_k}(t_i - t_k)} - \lambda_{s_i} \sum_{j=i+1}^{n} \left(c_{s_j} \prod_{k=1}^{j-1} \mathrm{e}^{-\lambda_{s_k}(t_j - t_k)} \right) \tag{14}$$

To find the maximum, we set this to zero and divide both sides by $\prod_{k=1}^{i} \mathrm{e}^{\lambda_{s_k} t_k}$:

$$0 = -V \cdot \lambda_{s_i} \prod_{j=1}^{n} \mathrm{e}^{-\lambda_{s_j} D} \prod_{j=i+1}^{n} \mathrm{e}^{\lambda_{s_j} t_j} + c_{s_i} \sum_{j=1}^{i-1} \lambda_{s_j} \prod_{k=1}^{i} \mathrm{e}^{-\lambda_{s_k} t_i} - \lambda_{s_i} \sum_{j=i+1}^{n} \left(c_{s_j} \prod_{k=1}^{j-1} \mathrm{e}^{-\lambda_{s_k} t_j} \prod_{k=i+1}^{j-1} \mathrm{e}^{\lambda_{s_k} t_k} \right) \tag{15}$$

[6] This is in terms of the number of arms.

[7] The exponential distribution is also a memoryless distribution, which leads to a compact solution. However, similar approaches can be applied to other distributions, or, when an analytical solution is intractable or impossible, time can be discretised and a solution can be found for arbitrary distributions. We will return to this in Section 5.3.3 and also perform experiments with other distributions in Section 6.1.2.

Here, we note that t_i is independent of any t_j, $j < i$, i.e., the invocation time of a provider does not depend on the invocation time of those already running. This is a result of the exponential function being memoryless, i.e., the probability of completing the task within the next time interval Δt is independent of when it was invoked. Hence, we can calculate each t_i by backward induction, starting with the last provider, n. The invocation time of this can be obtained directly by taking the derivative with respect to t_n (as in Eq. (15)):

$$t_n = D + \frac{\ln(c_{s_n} \cdot \sum_{j=1}^{n-1} \lambda_{s_j}) - \ln(V \cdot \lambda_{s_n})}{\sum_{j=1}^{n} \lambda_{s_j}} \tag{16}$$

Furthermore, we can obtain a simpler closed-form solution for the remaining invocation times by combining and manipulating the partial derivatives for t_i and t_{i+1}, resulting in:

$$\begin{aligned} &\frac{c_{s_i}}{\lambda_{s_i}} \sum_{j=1}^{i-1} \lambda_{s_j} \prod_{k=1}^{i-1} \mathrm{e}^{-\lambda_{s_k}(t_i - t_k)} - \sum_{j=i+1}^{n} \left(c_{s_j} \prod_{k=1}^{j-1} \mathrm{e}^{-\lambda_{s_k}(t_j - t_k)} \right) \\ &= \frac{c_{s_{i+1}}}{\lambda_{s_{i+1}}} \sum_{j=1}^{i} \lambda_{s_j} \prod_{k=1}^{i} \mathrm{e}^{-\lambda_{s_k}(t_{i+1} - t_k)} - \sum_{j=i+2}^{n} \left(c_{s_j} \prod_{k=1}^{j-1} \mathrm{e}^{-\lambda_{s_k}(t_j - t_k)} \right) \end{aligned} \tag{17}$$

Then, using algebraic manipulations, we isolate t_i, and derive an expression that is based solely on t_{i+1}:

$$t_i = t_{i+1} - \frac{1}{\sum_{j=1}^{i} \lambda_{s_j}} \ln\left(\frac{c_{s_{i+1}} \lambda_{s_i} \sum_{j=1}^{i+1} \lambda_{s_j}}{c_{s_i} \lambda_{s_{i+1}} \sum_{j=1}^{i-1} \lambda_{s_j}} \right) \tag{18}$$

Note that Eq. (18) is not well defined for t_1, but the optimal here is to set $t_1 = 0$. This is because the cost will be incurred in any case and any delays would only reduce its probability of success by the deadline. Furthermore, we note that the equations can yield negative values for some t_i, indicating that the optimal values lie outside the constraints of the problem (i.e., before the task can be started). In this case, as t_i does not influence the procurement times of later providers, the optimal choice is to set $t_i = 0$, i.e., the provider is invoked at the earliest possible time. Furthermore, the equations can sometimes yield inconsistent values, i.e., $t_i > D$ or $t_i > t_{i+1}$ for some i, but this only occurs when the ordering and/or the set of providers was non-optimal in the first place. Finally, we also note that the partial second derivatives are always negative, and since each variable is found uniquely one at a time using backward induction, the final result is optimal.

In the next section, we show how the optimal invocation times ρ_t^* can be found when service durations are perfectly correlated.

4.2.2. Perfectly correlated durations

We now consider optimal procurement strategies for scenarios where the only uncertainty stems from the difficulty of the task itself. As a result, the duration distributions are perfectly correlated and, as discussed in Section 3.3.2, when calculating the social welfare, this allows us to replace the union operator with a maximisation operator (see Eq. (13)). In this section, we will simplify the equation even further in the case of an optimal procurement strategy. For convenience, we will restate the equation in slightly different terms:

$$\overline{w}(\rho) = V \cdot G\Big(\max_{1 \leqslant i \leqslant n} q_{s_i}(D - t_i) \Big) - c_{s_1} - \sum_{i=2}^{n} c_{s_i} \cdot \Big[1 - G\Big(\max_{1 \leqslant j \leqslant i-1} q_{s_j}(t_i - t_j) \Big) \Big] \tag{19}$$

Now, the following properties enable us to considerably simplify the above equation:

Lemma 1. *Any optimal procurement strategy (in perfectly correlated settings), ρ^*, either has the following properties or can be converted into an equivalent (in terms of the expected social welfare) strategy having these properties*:

1. *The overall success probability only depends on the provider which is invoked last. Formally*:

$$\operatorname*{argmax}_{1 \leqslant i \leqslant n} q_{s_i}(D - t_i) = n$$

2. *The probability that provider s_i is invoked only depends on the provider which is invoked just before it, s_{i-1}. Formally, for any $i \in \{2, \ldots, n\}$*:

$$\operatorname*{argmax}_{1 \leqslant j \leqslant i-1} q_{s_j}(t_i - t_j) = i - 1$$

Proof. We prove both properties in turn. Suppose the first property does not hold, and there exists another provider s_k at position k in the schedule, so that $\operatorname{argmax}_{1 \leqslant i \leqslant n} q_{s_i}(D - t_i) = k$, where $k < n$. In that case, we can remove all providers

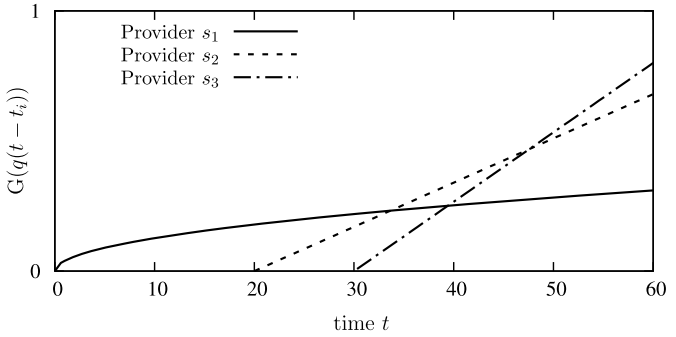

Fig. 3. Example showing the probability of success for each provider over time for a procurement strategy with 3 service providers, where $t_1 = 0$, $t_2 = 20$, and $t_3 = 30$. This procurement schedule satisfies the first property from Lemma 1, but not the second.

s_j, $j > k$ from the procurement strategy without adversely affecting the expected social welfare. To see this, note that removing these providers does not affect the first term in Eq. (19) (since this value is determined by provider s_k), nor does it affect the probability of invoking any of the providers up to and including the kth provider in the schedule. As a result of this manipulation, property (1) holds, since provider s_k will be the last provider in the schedule.

Now, assuming property (1) holds, we prove the second property using an inductive argument. Suppose this time that property (2) holds for all s_j, $j > i$ (i.e., all providers invoked after s_i), but not for provider s_i, $i \leqslant n$. That is, there exists a provider s_k, $k < i - 1$, such that $\operatorname{argmax}_{1 \leqslant j \leqslant i-1} q_{s_j}(t_i - t_j) = k$. In this case, we can remove provider s_{i-1} from the procurement schedule without negatively affecting Eq. (19). To see this, note that provider s_{i-1} does not affect the overall probability of success since this is determined by provider s_n (as per property (1)). Furthermore, s_{i-1} does not affect the invocation probability of any provider preceding it. Crucially, however, provider s_{i-1} also does not affect the invocation probability of provider s_i (since this is determined by provider s_k), nor does it affect any provider s_j with $j > i$ (since, by assumption, the invocation probability is determined by provider s_{j-1}). Now, by starting from the last provider, s_n, this elimination process can be repeated until property (2) holds. □

A graphical example to clarify these properties is depicted in Fig. 3. In this example, s_3 is invoked last and can also achieve the maximum overall probability of success by the deadline. Therefore, this example satisfies property (1) of Lemma 1. On the other hand, note that, at $t = 30$ when s_3 is invoked, the highest probability of success is determined by provider s_1 instead of s_2. This violates property (2) of Lemma 1. We can do better, therefore, by removing provider s_2 from the schedule, since it does not affect the overall probability of success, nor the invocation probability of any of the other providers. However, by removing provider s_2, we reduce the expected costs (hence, the strategy in Fig. 3 cannot be optimal).

Although there may exist multiple optimal strategies, due to Lemma 1 we can restrict our attention to those which satisfy properties (1) and (2). This way we can simplify Eq. (19) by removing the max operators, which results in:

$$\overline{w}(\rho^*) = V \cdot F_{s_n}(D - t_n) - c_{s_1} - \sum_{i=2}^{n} c_{s_i} \cdot \left[1 - F_{s_{i-1}}(t_i - t_{i-1})\right] \tag{20}$$

In what follows, we will use the simplified equation to find the optimal strategy. As in Section 4.2.1, we can state a number of necessary conditions for finding the optimal invocation times, given an appropriate ordering. We do this by setting $\partial \overline{w}(\rho^*)/\partial t_i = 0$, which results in:

$$0 = c_{s_i} \cdot f_{s_{i-1}}(t_i - t_{i-1}) - c_{s_{i+1}} \cdot f_{s_i}(t_{i+1} - t_i), \quad \text{where } 1 < i < n \tag{21}$$

$$0 = c_{s_n} \cdot f_{s_{n-1}}(t_n - t_{n-1}) - V \cdot f_{s_n}(D - t_n) \tag{22}$$

where $f_{s_i}(x) = \mathrm{d}F_{s_i}(x)/\mathrm{d}x$. As before, we note that it is always optimal to invoke the first provider at the earliest possible time and so $t_1 = 0$.

This gives us a system of simultaneous equations that can be solved for all t_i to yield the optimal invocation times. As in Section 4.2.1, we use the exponential distribution in the following to demonstrate how this can be done in practice. In more detail, we assume that $F_{s_i}(t) = 1 - \mathrm{e}^{-\lambda_{s_i} t}$ and $f_{s_i}(t) = \lambda_{s_i} \mathrm{e}^{-\lambda_{s_i} t}$. This arises, for example, if the problem difficulty Y is distributed according to an exponential distribution $G(y) = 1 - \mathrm{e}^{-y}$, and each provider has a quality of service function $q_{s_i}(t) = \lambda_{s_i} t$ (i.e., for each provider, the execution time depends linearly on the difficulty). Hence, $F_{s_i}(t) = G(q_{s_i}(t)) = 1 - \mathrm{e}^{-\lambda_{s_i} t}$. Moreover, for ease of notation, we let $\Delta t_i = t_i - t_{i-1}$ be the waiting time between invoking provider s_{i-1} and s_i (with $\Delta t_1 = 0$), allowing us to re-write Eqs. (21) and (22) as follows:

$$0 = c_{s_i} \lambda_{s_{i-1}} \mathrm{e}^{-\lambda_{s_{i-1}} \Delta t_i} - c_{s_{i+1}} \lambda_i \mathrm{e}^{-\lambda_{s_i} \Delta t_{i+1}}, \quad \text{where } 1 < i < n \tag{23}$$

$$0 = c_{s_n} \lambda_{s_{n-1}} \mathrm{e}^{-\lambda_{s_{n-1}} \Delta t_n} - V \lambda_{s_n} \mathrm{e}^{-\lambda_{s_n}(D - t_n)} \tag{24}$$

Rearranging this, we see that the following equalities hold:

$$c_{s_i}\lambda_{s_{i-1}}\mathrm{e}^{-\lambda_{s_{i-1}}\Delta t_i} = c_{s_j}\lambda_{s_{j-1}}\mathrm{e}^{-\lambda_{s_{j-1}}\Delta t_j}, \quad \text{where } 1 < i, j \leqslant n \tag{25}$$

$$c_{s_i}\lambda_{s_{i-1}}\mathrm{e}^{-\lambda_{s_{i-1}}\Delta t_i} = V\lambda_{s_n}\mathrm{e}^{-\lambda_{s_n}(D-t_n)}, \quad \text{where } 1 < i \leqslant n \tag{26}$$

Using Eq. (25), we can thus easily calculate any Δt_i, given another Δt_j (with $1 < j$):

$$\Delta t_i = \frac{\ln(\frac{c_{s_i}\lambda_{i-1}}{c_{s_j}\lambda_{s_{j-1}}}) + \lambda_{s_{j-1}}\Delta t_j}{\lambda_{s_{i-1}}} \tag{27}$$

Given this, we note $t_n = \sum_{i=2}^{n}\Delta t_i$ and use this to replace t_n in Eq. (26). This allows us to calculate Δt_n directly[8]:

$$c_{s_n}\lambda_{s_{n-1}}\mathrm{e}^{-\lambda_{s_{n-1}}\Delta t_n} = V\lambda_{s_n}\mathrm{e}^{-\lambda_{s_n}(D-\sum_{i=2}^{n-1}\frac{\ln(\frac{c_{s_i}\lambda_{s_{i-1}}}{c_{s_n}\lambda_{s_{n-1}}})+\lambda_{s_{n-1}}\Delta t_n}{\lambda_{s_{i-1}}}-\Delta t_n)} \tag{28}$$

Solving this for Δt_n, we obtain:

$$\Delta t_n = \frac{\ln(\frac{c_{s_n}\lambda_{s_{n-1}}}{V\lambda_{s_n}}) + \lambda_{s_n}D - \lambda_{s_n}\sum_{i=2}^{n-1}\frac{1}{\lambda_{s_{i-1}}}\ln(\frac{c_{s_i}\lambda_{s_{i-1}}}{c_{s_n}\lambda_{s_{n-1}}})}{\lambda_{s_n} + \lambda_{s_{n-1}} + \lambda_{s_n}\lambda_{s_{n-1}}\sum_{i=2}^{n-1}\frac{1}{\lambda_{s_{i-1}}}} \tag{29}$$

This allows us to calculate the waiting time for the last provider Δt_n, which we can then use to calculate all other waiting times using Eq. (27). Deriving the actual invocation time t_i of each provider is then trivial, i.e., $t_1 = 0$ and $t_i = \sum_{j=1}^{i}\Delta t_j$.

We now move on to the problem of finding the optimal ordering and subset of providers.

4.3. Optimal provider sequence

So far, the equations developed in Sections 4.2.1 and 4.2.2 allow us to efficiently calculate the optimal procurement times for a given sequence of service providers ρ_s^*. However, it is not obvious how to find this optimal set and ordering of providers. Related work on economic search that orders alternatives using reservation values or allocation indices, such as [56] or [17], does not directly apply to this case, due to the overlap of concurrently invoked providers. Furthermore, our problem includes a fixed time constraint, by which the task has to be completed. Other greedy approaches that order services by increasing costs, decreasing rate parameters, the ratio of these, or approaches that first select providers who individually yield a higher expected utility, also do not always find optimal solutions. This is because it is often best to select cheaper, slower providers first and only invoke the more expensive and faster ones later, to ensure that the task is completed successfully. However, when the deadline of the task is particularly short, the consumer may be forced to immediately invoke the faster, expensive providers.

As a simple example of this, we consider a set of two providers, $M = \{1, 2\}$. The first is cheap and slow with $c_1 = 0.2$ and $\lambda_1 = 0.1$, while the second is expensive and fast with $c_2 = 5$ and $\lambda_2 = 10$. For the sake of this example, their durations are here independent. If we now assume that a consumer has a task T with deadline $D = 1.5$ and utility $V = 100$, the optimal procurement strategy is $\rho^* = \langle(1, 0), (2, 0.75)\rangle$. However, if we decrease the deadline slightly to $D = 1$, the optimal strategy becomes $\rho^* = \langle(2, 0), (1, 0.84)\rangle$, thereby reversing the order of invoked providers. This observation suggests that a simple greedy search for the optimal strategy is insufficient. However, using a brute-force search over all possible subsets and orderings is clearly infeasible when the number of providers rises beyond a handful, as the number of possible orderings for m providers is $\sum_{i=0}^{m}\binom{m}{i}\cdot i! = \sum_{i=0}^{m} m!/(m-i)!$.

Fortunately, it *is* possible to quickly obtain an optimal order of providers when we make certain realistic assumptions about the quality of service functions in the perfectly correlated setting. We describe these assumptions in more detail in Section 4.3.1. Then, we develop a generic optimal branch-and-bound algorithm in Section 4.3.2, describing specific adjustments for both scenarios with independent and perfectly correlated durations. As this algorithm significantly reduces the space of solutions that have to be searched, it copes well with larger problems with up to ten or twenty service providers. For situations where even this becomes infeasible (when there are dozens of providers or more), we also develop a fast greedy algorithm in Section 4.3.3.

4.3.1. Optimal order in perfectly correlated setting

In the perfectly correlated setting, it is often reasonable to make certain assumptions about the quality of service functions q_i of the providers. In particular, we note that in these settings, some providers will always be able to complete a more difficult task than others within the same time period. For example, if we consider two otherwise identical PCs with clock speeds of 2 GHz (provider 1) and 3 GHz (provider 2), then it can safely be assumed that, if both run for the same

[8] The same procedure can be used to calculate an arbitrary Δt_j. We pick Δt_n here, as it leads to a simpler expression.

amount of time, the maximum task difficulty that provider 2 will be able to complete will always be higher. We formalise this in the following.

Assumption 1. There exists a total (strict) ordering of providers $<_d$, where $i <_d j$ implies that $q_i(x) < q_j(x)$ for all $x \in \mathbb{R}^+$.[9]

We note that a wide range of functions satisfy this assumption, including linear and, more generally, any polynomial or exponential function, where, for each of the constants, the providers have the same order. It also applies to the functions we assumed in Section 4.2.2. This assumption now gives us an unambiguous ordering over the providers that must be preserved by an optimal sequence:

Lemma 2. *Given Assumption* 1, *for any two providers i and j in the optimal sequence* ρ_s^*, *if* $i <_d j$, *then i is invoked before j, i.e.,* $t_i^* < t_j^*$.

Proof. The proof is fairly straightforward and can be demonstrated by contradiction. Suppose that, in the optimal schedule, $t_i^* > t_j^*$ (i is invoked after j). By assumption, $q_i(x) < q_j(x)$ for all $x \geqslant 0$. Because q_j is increasing (see Section 3.3.1), $q_i(x) < q_j(y)$ for any $0 \leqslant x \leqslant y$. Let $x = t - t_i^*$, $y = t - t_j^*$, where t is the current time, then $q_i(t - t*_i) < q_j(t - t_j^*)$ for any $t \geqslant t_i^*$. This means that, at any point in time, any task that can be completed by provider i at time t can also be completed by provider j at time $t' < t$. Therefore, provider i is superfluous, and cannot be part of ρ_s^*, which contradicts the assumption. □

The above observation is useful since it considerably reduces the search space. We can further prune this by noting that it is never beneficial to invoke a slower, more expensive provider before a faster, cheaper provider. Intuitively, this holds because we can simply invoke the second provider immediately and completely discard the first. This leads to a higher overall success probability and reduces the expected cost, thereby increasing the social welfare. Hence, we can remove any providers that are dominated in this way, resulting in a sequence of providers ordered both by increasing speed (i.e., by $<_d$) and by increasing costs.

Although this now gives us an ordering for the optimal sequence, we are still left with the problem of finding the optimal *subset* of providers to invoke. Thus, in the following section, we describe a generic branch-and-bound algorithm that applies to all settings considered so far. In particular, this works for general settings with independent or perfectly correlated durations. In addition, when Assumption 1 applies, it uses the results from this section to speed up the search.

4.3.2. Generic branch-and-bound algorithm

In this section we introduce a branch-and-bound algorithm that can be used for settings with correlated as well as independent durations, in order to reduce the space of subsets and orderings that have to be searched. This algorithm iteratively partitions the set of solutions into smaller subsets. Each time this happens, the algorithm computes a lower and an upper bound for all solutions in a given subset, allowing it to quickly prune those subsets that cannot contain an optimal solution (i.e., those with an upper bound that is less than some lower bound found so far).

More specifically, our branch-and-bound algorithm is based on a number of general observations:

- Some providers are inherently unsuitable for a given problem and therefore many orderings containing these providers can be discarded. For example, if the consumer immediately invokes provider 1 with $c_1 = 9$ for a task with value $V = 10$, its profit will be at most 1 (no matter what other providers are invoked later). If it knows that using a different provider already promises a better utility than this, it can immediately discard all orderings that begin with provider 1.
- As more providers are added to the end of a given ordering, their addition often has increasingly diminishing returns, allowing us to discard some solutions with many providers. For example, if the ordering $\langle 1, 2, 3 \rangle$ already promises a success probability close to 1, it may not be necessary to consider all solutions that start with these three providers (such as $\langle 1, 2, 3, 4 \rangle$, $\langle 1, 2, 3, 5 \rangle$, $\langle 1, 2, 3, 4, 5 \rangle$, $\langle 1, 2, 3, 5, 4 \rangle$). This is because adding further providers to the end only increases the cost, but will not significantly improve the success probability.
- Some providers clearly dominate others. For example, if provider i is cheaper and faster than j, then j will never be invoked before i.[10] Hence, we can remove all orderings where j is before i.
- In the correlated case, the assumption described in Section 4.3.1 may allow us to find a unique ordering over all providers (by increasing cost and quality) that must hold in the optimal solution and it may also enable us to further discard a number of dominated providers. However, note that this only applies in the correlated case and not when durations are independent. Hence, in the following, we will sometimes consider the independent and correlated settings separately where applicable.

[9] For simplicity here we assume the ordering to be strict at all points in time, but this requirement can be weakened.

[10] Note that provider i is faster than j if $\forall x$: $F_i(x) \geqslant F_j(x)$.

Algorithm 1 Branch-and-bound algorithm.

```
1: ρ_s^* ← ⟨⟩                                          ▷ Best ordering found so far
2: u_lower ← 0                                         ▷ Best current lower bound
3: Q ← {ρ_s^*}                                         ▷ Unexpanded orderings
4: while Q ≠ ∅ do                                      ▷ More unexpanded?
5:     ρ_s ← argmax_{ρ_s ∈ Q} LOWER(ρ_s)               ▷ Pick best
6:     Q ← Q \ {ρ_s}                                   ▷ Remove ρ_s from Q
7:     P'_s ← EXPAND(ρ_s)                              ▷ Expand ρ_s
8:     for all ρ_s' ∈ P'_s do
9:         ŭ ← LOWER(ρ_s')                             ▷ Find lower bound
10:        û ← UPPER(ρ_s')                             ▷ Find upper bound
11:        if û > u_lower then                         ▷ Sufficient upper bound?
12:            Q ← Q ∪ {ρ_s'}                          ▷ Keep for future expansion
13:            if ŭ > u_lower then                     ▷ Better lower bound?
14:                ρ_s^* ← ρ_s'                        ▷ Keep as current best
15:                u_lower ← ŭ
16:            end if
17:        end if
18:    end for
19:    Q ← {x ∈ Q | UPPER(x) > u_lower}                ▷ Filter orderings
20: end while
21: return FINDTIMES(ρ_s^*)                            ▷ Return best strategy
```

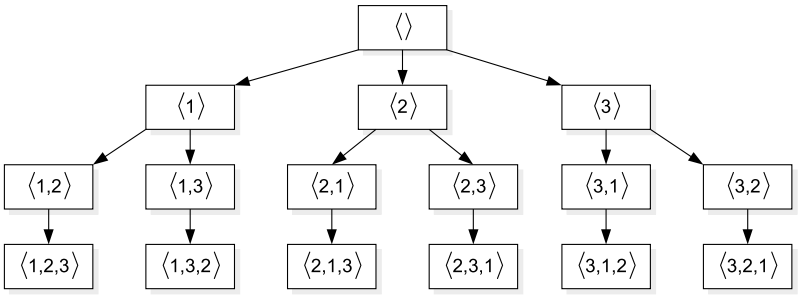

Fig. 4. Search tree that is explored by branch-and-bound algorithm (for $M = \{1, 2, 3\}$).

These intuitions are captured by the branch-and-bound technique used in Algorithm 1. In more detail, we begin with an empty ordering $\rho^*{}_s = \langle\rangle$ (line 1), and then repeatedly consider any new ordering that can be created by appending a single provider *to the end* of an existing ordering, thus exploring a search tree as exemplified by Fig. 4 (for $M = \{1, 2, 3\}$). We do this by keeping a queue of unexpanded orderings (Q in line 3). Then, at each iteration of the main loop (lines 4–20), we first select for expansion and remove from Q the ordering (lines 5 and 6) that promises the highest lower bound. Here, the function $\textsc{Lower}(\rho_s)$ is simply the utility of the ordering ρ_s, i.e., $\textsc{Lower}(\rho_s) = \overline{w}(\textsc{FindTimes}(\rho_s))$, where FINDTIMES returns the optimal procurement strategy using the equations from Sections 4.2.1 and 4.2.2. This trivially represents a strict lower bound for any ordering that starts with the providers given by ρ_s (i.e., in the search tree, this corresponds to ρ_s and all of its children).

Now, given the ordering to be expanded, ρ_s, we use the function $\textsc{Expand}(\rho_s)$ in line 7 to give us a new set of orderings, P'_s, each of which is created by adding a single unused provider to the end of ρ_s. In the search tree, this corresponds to generating the children of a particular ordering. For example, in Fig. 4, $\textsc{Expand}(\langle 1\rangle) = \{\langle 1, 2\rangle, \langle 1, 3\rangle\}$. As part of this function, we also immediately remove any orderings that are clearly infeasible. This depends on the setting as follows:

- When durations are **independent**, we remove all orderings where the last provider is dominated by any providers that are not part of that ordering. Here, provider j is dominated by provider i if the latter is faster and costs at most as much as j or if i is cheaper and as fast or faster than j.
- When durations are **perfectly correlated** and **can be ordered** (i.e., Assumption 1 holds), we remove all orderings where $j >_d i$ holds for the last provider i and another provider j that is already part of the ordering (where $>_d$ is defined as in Assumption 1). We also remove all orderings where $j >_d i \wedge c_i > c_j$ or $i \leqslant_d j \wedge j >_d i \wedge c_i \leqslant c_j$ holds for the last provider i and another provider j that is not part of the ordering. In other words, we always add providers that are faster than any previously invoked providers and never any providers where a better uninvoked alternative exists (i.e., cheaper and as least as fast, or faster and at most as expensive).[11] We can safely ignore the orderings that are removed

[11] As mentioned earlier, note that this filtering can be efficiently implemented by precomputing a list of all providers, ordered according to $<_d$, and then removing all providers where another provider is as expensive or cheaper later in the list. When running the $\textsc{Expand}(\rho_s)$ function, we then simply only add those providers that come later in the list than the last provider in ρ_s.

in this manner, because they, and any of their children in the search tree, are bound to be sub-optimal, as described in Section 4.3.1.

- When durations are **perfectly correlated** and **cannot be ordered** (i.e., Assumption 1 does not hold), we remove the same orderings as in the independent case.

For each of the new candidate solutions ρ_s' generated by the EXPAND(ρ_s) function, the algorithm then considers a lower bound, $\check{u}$, and an upper bound, $\hat{u}$, on the utility of ρ_s' and any of its children. As discussed above, the lower bound is simply the expected utility of ρ_s' when using the optimal invocation times. However, calculating an upper bound on the utility that could be achieved by adding any additional providers to the end of ρ_s' is not immediately obvious. Depending on the scenario, we do this in the following:

- If durations are **independent**, we let M' be the remaining service providers that are not in ρ_s'. If $M' = \emptyset$, then the upper bound is equal to the lower bound discussed above. Otherwise, we create a virtual service provider s_ρ with $c_{s_\rho} = \min_{i \in M'} c_i$ and $F_{s_\rho}(x) = 1 - \prod_{i \in M'}(1 - F_i(x))$.[12] This is based on the rationale that if any providers from M' are invoked in any order, their cost is bound to be at least c_{s_ρ} and their combined probability of success within any given time interval after invocation will never be higher than when immediately invoking all in parallel. With this reasoning, we obtain a new ordering ρ_s'' by appending s_ρ to ρ_s' and then calculate the upper bound as $\overline{w}$(FINDTIMES(ρ_s'')). If that is less than the lower bound, this indicates that it is not possible to achieve a higher utility by invoking further providers, and we can set the upper bound equal to the lower bound.
  We note here that occasionally the technique described in Section 4.2.1 will return inconsistent results when using these virtual providers. More specifically, it may be the case that $t_n < t_i$ for some $i < n$, which violates the ordering prescribed by ρ_s''. Briefly, this can occur when c_{s_ρ} is very small compared to the previously invoked providers, causing the second term in Eq. (18) to be negative. Normally, this would indicate that the ordering tested is suboptimal (clearly, we would want to invoke this provider as soon as possible, rather than at the end). However, in this case, s_ρ does not exist and we still need an upper bound without changing the initial ordering ρ_s'. Hence, when the result is inconsistent, we use a more conservative approach for calculating an upper bound. In more detail, we relax the problem and now assume that each provider is invoked in isolation and given the full D time units to complete the task. Furthermore, all selected providers are invoked in series until one is successful within its allocated time. Given that the order of the first invoked providers is already fixed by ρ_s', we then select the order of the remaining providers optimally by framing this relaxed problem as a simple economic search instance [56]. The expected utility of this is then a valid upper bound.
- If durations are **perfectly correlated** and **can be ordered**, we take a similar approach to calculating an upper bound by again creating a virtual service provider s_ρ that is guaranteed to perform better than any combination of the remaining providers. To do this, we let M' be the set of remaining service providers and set the cost of s_ρ to be the lowest cost in this set, i.e., $c_{s_\rho} = \min_{i \in M'} c_i$, and we set the duration function to be the best available in M', i.e., we set $F_{s_\rho}(x) = F_i(x)$, such that $i \in M'$ and $\forall j \in M' \setminus \{i\}$: $j <_d i$.[13] As before, in the special case where $M' = \emptyset$ or when the ordering with s_ρ yields a lower utility, we set the upper bound equal to the lower bound.
- If durations are **perfectly correlated** and **cannot be ordered**, we again have to take a slightly more conservative approach. Here we create a virtual service provider s_ρ that has the lowest cost in the set of remaining service providers, i.e., $c_{s_\rho} = \min_{i \in M'} c_i$. Since there may not be a single best duration function in that set, we then select $F_{s_\rho}(x) = \min_{i \in M'} F_i(x)$.

Now, given the lower and upper bounds for a particular candidate solution ρ_s' and all its children, we first compare its upper bound to the utility of the best ordering found so far (line 11). If it is not greater, then we cannot improve on the best solution so far by continuing to expand ρ_s'. Hence, we discard it from our search, thereby pruning a subset of the search-space. Otherwise, we add ρ_s' to Q for further expansion (line 12) and, if its utility is higher than the best ordering found so far, we update ρ_s^* and u_{lower} (lines 14 and 15).

At the end of each iteration, only unexpanded orderings with an upper bound that is higher than the currently highest lower bound are retained (line 19). This limits the size of Q (which we implemented using a priority queue), and also ensures that it is empty when all necessary orderings have been searched. When this happens, the best ordering and associated optimal times are returned (line 21). This final procurement strategy is optimal, because the algorithm searches all orderings, except for those that are known not to have a better expected utility than those already considered. Hence, the optimal ordering will never be discarded from the search.

In the following, we briefly illustrate the run-time behaviour of Algorithm 1 using an example problem instance. For this, we assume a task T with deadline $D = 2$ and value $V = 1$. There are three providers that can complete the task, as shown in Table 1, and we assume that their durations are independent. It is not obvious here whether the optimal strategy is to invoke the cheap provider 1 first, or immediately pick a more expensive and faster provider. To answer this, Fig. 5 illustrates how our algorithm explores the search space. More specifically, the figure shows all candidate solutions that are

[12] When exponential distributions are assumed, this is equivalent to creating a new provider with $\lambda_{s_\rho} = \sum_{i \in M'} \lambda_i$.

[13] When providers follow the exponential distribution, this is equivalent to setting $\lambda_{s_\rho} = \max_{i \in M'} \lambda_i$.

Table 1
Example service providers.

Provider (i)	Cost (c_i)	Rate (λ_i)
1	0.05	0.5
2	0.7	2.1
3	0.2	2

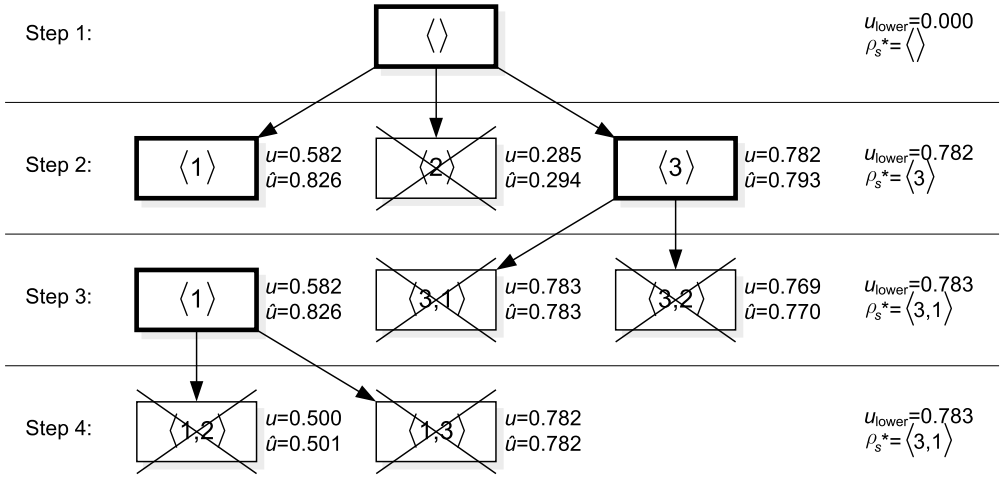

Fig. 5. Contents of Q during example run of Algorithm 1.

considered at each iteration of the main loop (lines 4–20). Here, the search nodes in bold signify orderings that are still members of Q at the end of the loop (after executing line 19), while the crossed-out nodes were considered during that iteration, but then subsequently removed, because their children do not offer improvements over the best solution found so far.

In more detail, the algorithm starts by considering the empty ordering $\rho_s = \langle\rangle$ (step 1), which is then expanded by appending each of the available providers to the end (step 2). This results in three new candidate solutions, for which the algorithm computes a lower ($\check{u}$) and an upper bound ($\hat{u}$), using the approaches described earlier in this section. As the ordering $\rho_s = \langle 2\rangle$ has an upper bound that is less than the lower bound of another solution, it is immediately discarded from the search (i.e., any ordering starting with provider 2 is subsequently ignored). As the other two orderings have overlapping bounds, they are both kept in Q for future expansion.

Next, the ordering $\rho_s = \langle 3\rangle$ is expanded, because it promises a higher lower bound than $\rho_s = \langle 1\rangle$ (step 3). One of its children, $\rho_s = \langle 3, 1\rangle$, is now the best ordering with $\check{u} = 0.783$, but neither of the two children can be further improved upon through expansion, so both are discarded. The only remaining node in Q, $\rho_s = \langle 1\rangle$, is then expanded (step 4), but neither child offers any possible improvement over 0.783, so both are discarded. As Q is now empty, the algorithm returns the best ordering found during its search, $\rho_s^* = \langle 3, 1\rangle$, which is the optimal.

However, while significantly reducing the search space in most realistic settings, this algorithm still searches for the optimal solution and may sometimes consider a large proportion of the entire search space. This may be the case, for example, when there are large numbers of highly similar providers and when the value of the task is very large in relation to the service costs. To address such scenarios, we introduce a fast heuristic approach in the following section.

4.3.3. Fast heuristic algorithm

Although we argued in the beginning of Section 4.3 that a greedy approach does not generally result in an optimal strategy, it can still achieve good results in practice and is more scalable than exhaustive approaches. Hence, we present such an algorithm that starts with an empty ordering and then greedily adds, removes or switches providers until a local optimum is reached (Algorithm 2). Intuitively, this algorithm benefits from selecting providers that offer a good trade-off between performance and cost. By also allowing providers to be removed or switched, it has some backtracking capabilities — thus an expensive but reliable provider can eventually be replaced by many cheap and unreliable providers that individually do not yield a high expected utility, but in combination result in a better strategy. Also, while the invocation order is generated greedily, the algorithm uses the results from Section 4.2 to efficiently calculate the *optimal* invocation times for a given ordering, thereby combining a greedy heuristic approach with optimal solution techniques.

Algorithm 2 Fast heuristic algorithm.

```
1: ρ_s^* ← ⟨⟩                                          ▷ Best ordering found so far
2: u^* ← 0                                             ▷ Best current utility
3: done ← false                                        ▷ Reached local optimum?
4: while done = false do
5:     P'_s ← GenerateNeighbours(ρ_s^*)                ▷ Generate neighbours of ρ_s^*
6:     ρ'_s ← argmax_{ρ_s ∈ P'_s} w̄(FindTimes(ρ_s))    ▷ Select best neighbour
7:     if w̄(FindTimes(ρ'_s)) > u^* then                ▷ If neighbour is better than ρ_s^*...
8:         ρ_s^* ← ρ'_s                                ▷ ...use as next candidate
9:         u^* ← w̄(FindTimes(ρ'_s))
10:    else
11:        done ← true                                 ▷ ...otherwise stop search
12:    end if
13: end while
14: return FindTimes(ρ_s^*)                            ▷ Return best strategy
```

In more detail, the algorithm uses a function, GenerateNeighbours(ρ_s^*) in line 5, which, given a current ordering ρ_s^*, returns all possible orderings that can be obtained by the following three actions: (1) selecting a provider x which is currently not in ρ_s^* and adding it to ρ_s^* at position $i \in \{1, 2, \ldots, n+1\}$ (shifting other providers as necessary), (2) selecting a provider s_i in ρ_s and removing it, or (3) selecting two providers s_i and s_j in ρ_s and swapping their positions. Out of these, the best neighbour is chosen and this continues until the algorithm cannot find another better ordering. In this case, the current best is returned.

While we have so far assumed that the consumer has complete information about the costs and duration functions of the providers, this is rarely available to the consumer in practice. Clearly, this is a significant obstacle, because our approach requires this information to make informed procurement decisions. For this reason, we show in the following section how it can be extended for settings where providers have private information about their services.

5. Using mechanism design to incentivise truthfulness in providers

So far, we have assumed that the consumer is provided with all the information it requires in order to compute the optimal procurement strategy. In practice, however, information about a provider's duration functions and its cost of executing a task may not be publicly available, and instead may be closely guarded private information. When asked, a provider may refuse to reveal this information, or may lie about it, for example, by claiming a higher cost or by misrepresenting its duration function in order to manipulate the final procurement strategy in its own favour. Since costs may be idiosyncratic and the duration functions are probabilistic, it can be very difficult for the consumer to verify whether the information revealed by a service provider is reliable.

To address this problem, we describe a number of *procurement mechanisms*, which, by introducing direct competition between the providers, provide incentives for them to truthfully reveal their private information to the consumer, allowing the consumer to be confident that it really has the best procurement strategy. We consider several mechanisms here, in order to cover a range of realistic scenarios, including where only some of the information about providers is private, and we also present alternative mechanisms that allow the consumer to trade-off computational and information requirements with the solution quality (see Table 2 at the end of Section 5.1 for a full summary of our proposed mechanisms and their specific properties).

We start this section by introducing key concepts from the field of mechanism design and by clarifying our notation. We then look at designing mechanisms which are suitable for settings where the duration probabilities of all providers is public knowledge, but the cost of each provider must be elicited. We end the section by investigating what is possible if both duration and cost information is privately held by the consumers.

5.1. Preliminaries

Mechanism design, a sub-field of game theory, studies how to design protocols for self-interested agents, so as to ensure that certain desirable properties are achieved. A mechanism specifies three aspects of a protocol: (1) the possible actions that the self-interested agents may take, (2) an outcome function which selects a particular outcome given what actions agents have taken, and (3) a transfer function that determines a payment to each of the agents, conditional on their actions. Ideally, a desirable outcome from the perspective of the mechanism designer will arise when agents are taking actions which are in equilibrium. While in general there need be no restrictions on the mechanism, in this paper, we will restrict our focus to *direct mechanisms*. A direct mechanism is one where the possible actions of the agents are restricted so that they are only allowed to reveal their private information, instead of taking arbitrary actions. This restriction is often made in the literature, since the *Revelation Principle* states that if a mechanism results in a desired outcome, then there exists a direct mechanism which will also result in the same outcome [34].

In our proposed mechanisms, the service providers are asked to reveal their private information, but we make no assumption that the information that they reveal is their true information. In particular, we let c_i and F_i be the cost and duration functions of provider i, and denote the cost and duration function revealed to the mechanism by $\hat{c}_i$ and $\hat{F}_i$. We

let $\hat{c} = \langle \hat{c}_1, \ldots, \hat{c}_m \rangle$ be the reported costs of all service providers, and define $\hat{F} = \langle \hat{F}_1, \ldots, \hat{F}_m \rangle$ analogously. As is standard, we use the notation $\hat{c}_{-i} = \langle \hat{c}_1, \ldots, \hat{c}_{i-1}, \hat{c}_{i+1}, \ldots, \hat{c}_m \rangle$ to denote all cost reports except from provider i (and $\hat{F}_{-i}$ is, again, defined in a similar manner). Thus, we sometimes write $\hat{c} = \langle \hat{c}_i, \hat{c}_{-i} \rangle$ and $\hat{F} = \langle \hat{F}_i, \hat{F}_{-i} \rangle$.

Given the information announced by the service providers, $\hat{c}$ and $\hat{F}$, it is possible to evaluate different procurement strategies. For example, we let $\overline{w}(\rho|\hat{c}, \hat{F})$ denote the expected social welfare of procurement strategy ρ given the reports of the providers, and $\overline{w}(\rho|c, F)$ is the true expected social welfare. Similarly, $\overline{w}(\rho|\hat{c}_{-i}, \hat{F}_{-i})$ is the expected social welfare of procurement strategy ρ if service provider i had never existed (and thus could not be part of the procurement strategy). The optimal procurement strategy, given $\hat{c}$ and $\hat{F}$, is $\rho^*(\hat{c}, \hat{F}) = \operatorname{argmax}_{\rho \in \mathcal{P}} \overline{w}(\rho|\hat{c}, \hat{F})$, and we use $\rho^*(\hat{c}_{-i}, \hat{F}_{-i}) = \operatorname{argmax}_{\rho \in \mathcal{P}} \overline{w}(\rho|\hat{c}_{-i}, \hat{F}_{-i})$ to denote the optimal procurement strategy without the presence of agent i. When the expected welfare and optimal procurement strategy are computed based on the same information, we will typically use the abbreviated notation $\overline{w}(\rho^*(\hat{c}, \hat{F})) = \overline{w}(\rho^*(\hat{c}, \hat{F})|\hat{c}, \hat{F})$. Similarly, $\overline{w}(\rho^*(c, F)) = \overline{w}(\rho^*(c, F)|c, F)$. Once the service providers report $\hat{c}$ and $\hat{F}$, the mechanism selects an outcome. In particular, in the rest of this section, we assume that the mechanism selects $\rho^*(\hat{c}, \hat{F})$ and then specifies transfer functions, τ_i, for each service provider. We currently do not specify τ_i, but will instantiate them later in this section, depending on what properties we wish to achieve.

There are several properties we desire for our mechanisms. First, the service providers cannot be forced to participate, and instead our mechanisms must be designed so that the service providers voluntarily participate. That is, we are interested in mechanisms that are *individually rational*.

Definition 6 *(Individual rationality).* A mechanism is (*ex-post*) *individually rational*, if, for any realisation of costs and duration distributions c and F, and for all $i \in M$, the following holds:

$$\bar{u}_i\big(\rho^*(c, F)\big) \geqslant 0.$$

If the utility of the providers is furthermore non-negative for any realisation of the completion time, $X_{\rho^*(c,F)}$ (i.e., even after execution of the procurement strategy), then the mechanism is called *post-execution* individually rational.

Second, we wish to design mechanisms which provide incentives so that the service providers are made best off when they reveal their costs and duration probabilities truthfully. In particular, when possible, we are interested in designing mechanisms where providers are best off revealing their true information, no matter what information the other providers reveal.

Definition 7 *(Incentive compatibility in dominant strategies).* A mechanism is *incentive compatible in dominant strategies*, if, for each provider i with c_i and F_i, and for all possible declarations by others, $\hat{c}_{-i}$ and $\hat{F}_{-i}$, and for all $\hat{c}_i \neq c_i$ and $\hat{F}_i \neq F_i$,

$$\bar{u}_i\big(\rho^*\big(\langle c_i, \hat{c}_{-i}\rangle, \langle F_i, \hat{F}_{-i}\rangle\big)\big) \geqslant \bar{u}_i\big(\rho^*\big(\langle \hat{c}_i, \hat{c}_{-i}\rangle, \langle \hat{F}_i, \hat{F}_{-i}\rangle\big)\big)$$

Incentive compatibility in dominant strategies is sometimes called *strategy-proofness* and we will use these terms interchangeably. If a mechanism is strategy-proof, then all service providers maximise their own expected utility by truthfully announcing their costs and duration functions. That is, $\hat{c}_i = c_i$ and $\hat{F}_i = F_i$ for all i. Since the mechanism selects the procurement strategy which maximises the social welfare given the reports of the service providers, by using an incentive compatible mechanism, the consumer can ensure that it is being provided with appropriate information from the providers, and is thus selecting the best procurement strategy.

Now, one prominent example of a mechanism that is incentive compatible in many settings is the well-known Vickrey–Clarke–Groves (VCG) mechanism [34]. This mechanism selects the outcome that maximises the social welfare and pays every agent its marginal contribution to the overall system. This leads to several desirable properties. Specifically, in private information settings, where the utility of an outcome to a particular agent is only determined by its own type, the VCG mechanism is incentive compatible in dominant strategies and it is individually rational. Furthermore, in certain settings, VCG is in fact the only incentive compatible mechanism possible [29].[14] As VCG is of key importance in the mechanism design literature and as it fulfills incentive compatibility, we base some of our mechanisms on it.

In the rest of this section, we study different scenarios and determine what the appropriate mechanisms are for these settings. We first assume that the duration probability information is publicly known and study what incentives must exist in order to ensure that the service providers willingly reveal their costs. We then investigate what happens if both the cost and duration probabilities are privately held. Table 2 provides a comprehensive summary of our results and can be used as an aid to choose the most appropriate mechanism for a given setting. The table indicates whether the mechanism handles only private information about costs, or both about costs and distributions, what properties distinguish each mechanism[15]

[14] This result does not directly apply here, as it assumes agents may have arbitrary utilities for outcomes (while our work assumes a restricted domain that arises from the service procurement scenario).

[15] Note that some concepts, such as ex-post incentive compatibility or the poly-time approximation algorithm are introduced and discussed later in the relevant sections.

Table 2
Summary of procurement mechanisms. Here, IC stands for *incentive compatible*, in DS means that it holds in *dominant strategies* and IR abbreviates *individually rational*.

Mechanism	Private		Properties	Solution algorithms
	Costs c_i	Dist. F_i		
Marginal contribution	✓	×	IC (in DS) IR (in expectation) Optimal welfare Incentive to de-commit	Optimal
Uniform pricing	✓	×	IC (in DS) IR (post-execution) Suboptimal welfare Requires parameter k	Optimal or greedy
Discriminatory pricing	✓	×	IC (in DS) IR (post-execution) Suboptimal welfare No parameter required	Optimal or greedy
Execution-Contingent VCG	✓	✓	IC (ex-post) IR (in expectation) Optimal welfare	Optimal
Approximate Execution-Contingent VCG	✓	✓	IC (ex-post) IR (in expectation) Suboptimal welfare	Poly-time approximation

and whether the mechanism requires an optimal solution algorithm (that scales to at most dozens of providers) or can be solved using a greedy heuristic or approximation (that scales to thousands).

5.2. Known duration probabilities, unknown costs

We initially assume that the duration probability functions, F_i, are publicly known, but that the cost functions of each service provider, c_i, are private.[16] While each provider is free to report any cost, $\hat{c}_i$, it wishes, we show that it is possible to design mechanisms for this setting, such that each provider maximises its expected utility by truthfully reporting its costs, that is $\hat{c}_i = c_i$.

5.2.1. Marginal contribution mechanism

Since we assume that F_i is already known, each provider is only asked to report a cost, $\hat{c}_i$. Given this, it is possible to adapt the standard VCG mechanism, discussed earlier, to this setting. More formally, using the reported costs and the known duration functions, the consumer finds the optimal procurement strategy, $\rho^*(\hat{c}, F)$. Then, before executing $\rho^*(\hat{c}, F)$, the consumer computes and pays each service provider $i \in M$ a transfer

$$\tau_i = \overline{w}_{-i}\big(\rho^*(\hat{c}, F)\big) - \overline{w}\big(\rho^*(\hat{c}_{-i}, F_{-i})\big). \tag{30}$$

The second term of the transfer, $\overline{w}(\rho^*(\hat{c}_{-i}, F_{-i}))$, is the expected social welfare of the optimal procurement strategy if provider i did not exist. The first term of the transfer, $\overline{w}_{-i}(\rho^*(\hat{c}, F))$, is the expected social welfare obtained by the optimal procurement strategy $\rho^*(\hat{c}, F)$, *excluding the reported cost of provider* i. That is

$$\overline{w}_{-i}\big(\rho^*(\hat{c}, F)\big) = V \cdot \text{Prob}(X_{\rho^*(\hat{c},F)} \leqslant D) - \sum_{j=1, j\neq i}^{n} \hat{c}_{s_j} \cdot \big(1 - \text{Prob}(X_{\rho^*(\hat{c},F)} \leqslant t_j)\big) \tag{31}$$

We emphasise that when computing $\overline{w}_{-i}(\rho^*(\hat{c}, F))$, only provider i's cost is ignored, but it is not removed completely from the social welfare. In particular, provider i's existence in the procurement strategy may affect the probability of success and therefore the consumer's utility, as well as that of other providers, since it may influence whether or not they are asked to attempt a task or not. However, if provider i was never a candidate for procurement in $\rho^*(\hat{c}, F)$, then its existence has no impact on the social welfare, and therefore $\overline{w}(\rho^*(\hat{c}_{-i}, F_{-i})) = \overline{w}_{-i}(\rho^*(\hat{c}, F))$.

By defining the transfers for each service provider i as was done in Eq. (30), it is straightforward to show that service provider i maximises its expected utility by truthfully reporting $\hat{c}_i = c_i$. Let $\overline{u}_i(\rho^*(\langle \hat{c}_i, \hat{c}_{-i}\rangle, F)|c_i)$ be service provider i's expected utility when all other service providers report $\hat{c}_{-i}$, provider i reports $\hat{c}_i$ and its actual cost is c_i. Then:

[16] The duration functions may be obtained from past or shared experiences, for example from using a trust or reputation system, or simply given by the provider.

$$\begin{aligned}\bar{u}_i\big(\rho^*(\langle\hat{c}_i,\hat{c}_{-i}\rangle,F)\big|c_i\big) &= \tau_i - c_i\cdot\big(1-\text{Prob}(X_{\rho^*(\langle\hat{c}_i,\hat{c}_{-i}\rangle,F)}\leqslant t_i)\big)\\ &= \overline{w}_{-i}\big(\rho^*\big(\langle\hat{c}_i,\hat{c}_{-i}\rangle,F\big)\big|\hat{c}_{-i},F\big)-\overline{w}\big(\rho^*(\hat{c}_{-i},F_{-i})\big)-c_i\cdot\big(1-\text{Prob}(X_{\rho^*(\langle\hat{c}_i,\hat{c}_{-i}\rangle,F)}\leqslant t_i)\big)\\ &= \overline{w}\big(\rho^*\big(\langle\hat{c}_i,\hat{c}_{-i}\rangle,F\big)\big|\langle c_i,\hat{c}_{-i}\rangle,F\big)-\overline{w}\big(\rho^*(\hat{c}_{-i},F_{-i})\big)\end{aligned}\tag{32}$$

Since $\rho^*(\hat{c},F)$ is, by definition, the procurement strategy which maximises $\overline{w}$ given reports $\hat{c}$, provider i can optimise the first term from its perspective by reporting $\hat{c}_i=c_i$. As for the second term in the utility function, $\overline{w}(\rho^*(\hat{c}_{-i},F_{-i}))$, provider i has no influence on this term, no matter what the revealed cost, since this is based on a procurement strategy where provider i is excluded. Therefore, the service provider is best off revealing its true cost if it wishes to maximise its expected utility. That is, the mechanism is incentive-compatible in dominant strategies (i.e., strategy-proof).

This is not surprising, as the mechanism just presented is simply an application of the VCG mechanism to our procurement setting and therefore inherits its desirable properties. However, while this marginal contribution mechanism is both strategy-proof and (ex-post) individually rational, it has two weaknesses. First, the mechanism is individually rational in *expectation* only, and not *post-execution* individually rational. Thus, for particular instances, upon executing the procurement strategy, a provider may end up with negative utility, as illustrated in Example 1 below.

Example 1. Let $V=10$, and assume there are two providers with costs $c_1=c_2=5$. For simplicity, assume that there are two time steps of interest before the deadline, $t_1=0$ and t_2. Furthermore, let $F_1(D)=F_1(D-t_2)=0.9$ and $F_2(D)=F_2(D-t_2)=0.8$. In other words, the providers each have a 90% and 80% probability of succeeding, respectively, no matter at which time step they are invoked. In this case, it is optimal to invoke provider 1 on the first time step, and, if this fails, to invoke provider 2 on the second time step. The expected social welfare functions for different configurations are given by:

$$\begin{aligned}&\overline{w}\big(\rho^*(c,F)\big)=V\cdot\big(1-(1-0.9)\cdot(1-0.8)\big)-c_1-(1-0.9)\cdot c_2=4.3\\ &\overline{w}_{-1}\big(\rho^*(c,F)\big)=V\cdot\big(1-(1-0.9)\cdot(1-0.8)\big)-(1-0.9)\cdot c_2=9.3\\ &\overline{w}_{-2}\big(\rho^*(c,F)\big)=V\cdot\big(1-(1-0.9)\cdot(1-0.8)\big)-c_1=4.8\\ &\overline{w}\big(\rho^*(c_{-1},F_{-1})\big)=V\cdot 0.8-c_2=3\\ &\overline{w}\big(\rho^*(c_{-2},F_{-2})\big)=V\cdot 0.9-c_1=4\end{aligned}$$

And the transfers and expected utilities of the providers are:

$$\begin{aligned}&\tau_1=\overline{w}_{-1}\big(\rho^*(c,F)\big)-\overline{w}\big(\rho^*(c_{-1},F_{-1})\big)=6.3\\ &\tau_2=\overline{w}_{-2}\big(\rho^*(c,F)\big)-\overline{w}\big(\rho^*(c_{-2},F_{-2})\big)=0.8\\ &\bar{u}_1=\tau_1-c_1=1.3\\ &\bar{u}_2=\tau_2-(1-0.9)\cdot c_2=0.3\end{aligned}$$

Note that if provider 2 is never invoked, then its *post-execution* utility is 0.8. If, however, provider 2 is invoked, then its *post-execution* utility is -4.2.

In this example, note that, once the procurement strategy is executed, if provider 2 is invoked, then it has an incentive to *de-commit*, that is, to refuse to execute the task and to forego the transfers from the consumer. This holds because, even though the transfers are positive and exceed the *expected costs*, they are less than the *actual* costs incurred upon execution. Therefore, unless the consumer can otherwise enforce the schedule, for example through the use of de-commitment penalties, the marginal contribution mechanism may fail in practice.[17]

The second weakness which arises with respect to the marginal contribution mechanism is the computational burden it places on the consumer. The consumer must compute the optimal procurement strategy when considering all providers as candidates, and then the optimal procurement strategy as each provider is removed from consideration. This problem is further exacerbated by the fact that the consumer has limited computational power to start with; that is why it is procuring services from the providers. While a heuristic algorithm was proposed for handling settings with large number of providers (Section 4.3.3), it has been well established that many mechanisms, including VCG mechanisms like ours, may not be incentive compatible if the outcome selected is sub-optimal and does not maximise social welfare [36].[18] Therefore, heuristics and approximation algorithms must be carefully designed in order to ensure that the mechanism maintains the desired strategic properties. To this end, in the rest of this section, we investigate alternative mechanisms which address both the computational overhead and post-execution individual rationality, while maintaining incentive compatibility. These

[17] Another approach is to require providers to place a deposit, which should be at least as high as the highest cost, but this may not be desirable, since this may discourage providers from participating.

[18] Intuitively, this is because an agent can misreport its information in order to try and manipulate the approximation in its favour.

mechanisms vary in their information requirements and are therefore applicable in different settings, which we will examine in more detail in Section 6.

5.2.2. Uniform pricing mechanism

In the *uniform pricing mechanism*, the consumer first publicly announces an integer k, where $1 \leqslant k < m$. Then, as before, each provider $i \in M$ reports a cost, $\hat{c}_i$. Without loss of generality, we assume that $\hat{c}_i \leqslant \hat{c}_{i+1}$. We define $\mathcal{K} = \{i\colon\ i \in M \text{ and } \hat{c}_i < \hat{c}_{k+1}\}$ to be the set of k providers with the k lowest reported costs. This set determines the *candidate providers* and any provider in $M \setminus \mathcal{K}$ will not be considered when computing the procurement strategy. Let $F_{\mathcal{K}} = \langle F_1, \ldots, F_k \rangle$ be a vector which specifies the duration functions of each provider in $\mathcal{K}$, $\hat{c}_{\mathcal{K}} = \langle \hat{c}_1, \ldots, \hat{c}_k \rangle$ be a vector which specifies the reported costs for each provider in $\mathcal{K}$ and $\hat{c}^u_{\mathcal{K}} = \langle \hat{c}_{k+1}, \ldots, \hat{c}_{k+1} \rangle$ be a vector which, for each provider in $\mathcal{K}$, *replaces their announced cost,* $\hat{c}_i$*, with the lowest cost amongst all providers not in* $\mathcal{K}$*, that is,* $\hat{c}_{k+1}$.

Given the providers in $\mathcal{K}$ and their reported costs and publicly known duration functions, the consumer finds the procurement strategy which maximises social welfare while using only providers in $\mathcal{K}$ and assuming that their costs are all $\hat{c}_{k+1}$. That is,

$$\rho^*\big(\hat{c}^u_{\mathcal{K}}, F_{\mathcal{K}}\big) = \operatorname*{argmax}_{\rho \in \mathcal{P}:\ s_i \in \mathcal{K}} \overline{w}\big(\rho | \hat{c}^u_{\mathcal{K}}, F_{\mathcal{K}}\big).$$

Once the procurement strategy, $\rho^*(\hat{c}^u_{\mathcal{K}}, F_{\mathcal{K}})$, is found, the consumer executes the strategy and records a set $\mathcal{I}_{\mathcal{K}} \subseteq \mathcal{K}$ which comprises the providers in $\mathcal{K}$ that are actually invoked when using procurement strategy $\rho^*(\hat{c}^u_{\mathcal{K}}, F_{\mathcal{K}})$ (see also Definition 4 in Section 3.2). Only after the procurement strategy has been executed are the transfers to the providers determined as follows:

$$\tau_i = \begin{cases} \hat{c}_{k+1} & \text{if } i \in \mathcal{I}_{\mathcal{K}} \\ 0 & \text{if } i \in M \setminus \mathcal{I}_{\mathcal{K}} \end{cases} \tag{33}$$

We emphasise that the transfer τ_i is *conditional on the outcome of* $\rho^*(\hat{c}^u_{\mathcal{K}}, F_{\mathcal{K}})$. That is, a non-zero transfer to provider i only occurs if $i \in \mathcal{K}$ and i is actually invoked by the procurement strategy. Otherwise, a provider receives no transfer (but also incurs no cost).

There are several computational advantages of the uniform pricing mechanism, compared to the marginal contribution mechanism described in the previous section. First, depending on the k chosen by the consumer, the set of candidate providers may be significantly smaller than the entire pool of possible service providers. Secondly, the transfers of the providers are straightforward to compute, since they merely require that the consumer is able to sort the service providers by the reported costs. Thirdly, in some settings, such as when the duration probability distributions can be ordered (e.g., as described in Section 4.3.1), the problem of finding the optimal procurement strategy reduces down to selecting the best provider with the highest probability of completing the task within the deadline.

Next, we show formally that the uniform pricing mechanism is both post-execution individually rational and incentive compatible in dominant strategies.

Theorem 1. *Let M be the set of service providers, $|M| = m$. For any k such that $1 \leqslant k < m$, the uniform pricing mechanism is*:

- *Incentive compatible in dominant strategies, and*
- *Post-execution individually rational.*

Proof. We start by proving that the mechanism is incentive compatible. Assume all other providers report $\hat{c}_{-i}$ and that provider i's true cost is c_i. Assume that $c_i \geqslant \hat{c}_{k+1}$. Now, if provider i truthfully revealed c_i, then it would not belong to $\mathcal{K}$. Thus, it would not be part of the procurement strategy and thus it would incur no cost and receive no transfer. That is, its expected utility would be 0. If $\hat{c}_i > \hat{c}_{k+1}$, then provider i would still have zero utility since it would still not be a member of $\mathcal{K}$. If $\hat{c}_i < \hat{c}_{k+1} \leqslant c_i$, then $i \in \mathcal{K}$. If i is invoked, then it would incur cost c_i but only receive transfer $\hat{c}_{k+1}$, resulting in utility $\hat{c}_{k+1} - c_i \leqslant 0$. Thus, if provider i's true cost is greater than $\hat{c}_{k+1}$, then it cannot improve its utility by misreporting its cost. Now assume that $c_i < \hat{c}_{k+1}$. If invoked, then provider i would receive transfer $\hat{c}_{k+1}$ and incur cost c_i, resulting in utility $\hat{c}_{k+1} - c_i \geqslant 0$. If provider i reports any cost $\hat{c}_i$ which is less than $\hat{c}_{k+1}$, then it will receive the same utility as if it told the truth about its cost. If the provider reports $\hat{c}_i > \hat{c}_{k+1}$, then it would no longer be a candidate provider and would have utility equal to zero. Thus, again, provider i cannot improve its utility by misreporting its cost. This holds for any provider and thus the mechanism is incentive compatible in dominant strategies, or strategy-proof.

We now show that the mechanism is post-execution individually rational. Assume that provider i was *not* invoked during execution of the procurement strategy. Then $\tau_i = 0$, but it also incurred no cost. Therefore, $u_i = 0$. Now assume that provider i, with true cost c_i, was invoked. The transfer it receives is $\tau_i = \hat{c}_{k+1}$, but since i was invoked, then it must have been a member of $\mathcal{K}$, which means that $c_i \leqslant \hat{c}_{k+1}$ by definition of $\mathcal{K}$. Therefore

$$u_i = \tau_i - c_i = \hat{c}_{k+1} - c_i \geqslant 0. \qquad \square$$

While our uniform pricing mechanism overcomes the problems we highlighted with respect to the marginal contribution mechanism, it still has some key limitations. First, the initial stage, where $\mathcal{K}$ is chosen, depends only on the reported costs of the providers and ignores the duration probabilities, possibly leading to a situation where expensive, but very fast, providers are excluded. Second, the parameter k is a key part of the mechanism, and must be announced before any provider reveals cost information. To set k optimally requires *a priori* information about the distribution of the costs, which might be difficult to obtain.[19] To address this last problem, we introduce two variations of this mechanism.

5.2.3. Discriminatory pricing mechanisms

As mentioned in the previous section, one possible problem with our uniform pricing mechanism is that its effectiveness relies heavily on the choice of an appropriate value of k. In this section, we describe two mechanisms which no longer rely on the consumer setting the parameter appropriately, and are also *discriminatory* in that different providers may receive different transfers when invoked. These two mechanisms vary slightly in the way candidate providers are chosen and how transfers are determined, and, as we will note later, we have selected them here as two representative examples of a more general class of mechanisms.

In the *Pairing mechanism*, each provider $i \in M$ reports a cost $\hat{c}_i$. Then, all providers are *randomly* paired with another provider (and if $|M| = m$ is odd, then a single triplet is formed). For each pair, (i, j), if $\hat{c}_i < \hat{c}_j$ then i is put into candidate set $\mathcal{K}$ and its *paired cost*, $\hat{c}_i^p$, is set so that $\hat{c}_i^p = \hat{c}_j$. This procedure results in a set $\mathcal{K}$ such that $|\mathcal{K}| = \lfloor m/2 \rfloor$ and we let $\hat{c}_{\mathcal{K}}^p$ denote the vector specifying the paired cost of each provider in $\mathcal{K}$. Given $\mathcal{K}$, the consumer computes the optimal procurement strategy, $\rho^*(\hat{c}_{\mathcal{K}}^p, F_{\mathcal{K}})$, restricting itself to providers in $\mathcal{K}$ and using the costs from $\hat{c}_{\mathcal{K}}^p$:

$$\rho^*\big(\hat{c}_{\mathcal{K}}^p, F_{\mathcal{K}}\big) = \operatorname*{argmax}_{\rho \in \mathcal{P}:\, s_i \in \mathcal{K}} \overline{w}\big(\rho \big| \hat{c}_{\mathcal{K}}^p, F_{\mathcal{K}}\big).$$

After $\rho^*(\hat{c}_{\mathcal{K}}^p, F_{\mathcal{K}})$ is executed, transfers are determined for each provider. As in the previous section, let $\mathcal{I}_{\mathcal{K}} \subseteq \mathcal{K}$ denote the set of providers in $\mathcal{K}$ actually invoked when $\rho^*(\hat{c}_{\mathcal{K}}^p, F_K)$ was executed. Then

$$\tau_i = \begin{cases} \hat{c}_i^p & \text{if } i \in \mathcal{I}_{\mathcal{K}} \\ 0 & \text{if } i \in M \setminus \mathcal{I}_{\mathcal{K}} \end{cases} \tag{34}$$

Our *Halving* mechanism only differs from the Pairing mechanism in the way that it selects candidate service providers for $\mathcal{K}$. As before, all service providers are asked to report a cost, $\hat{c}_i$. Then $\lfloor m/2 \rfloor$ providers are randomly selected and put into a set $\mathcal{G}$. All providers in $M \setminus \mathcal{G}$ are paired together at random and the set $\mathcal{K}$ is created as described in the Pairing mechanism, with each provider $i \in \mathcal{K}$ having paired cost $\hat{c}_i^p$, determined as described in the Pairing mechanism. From $\mathcal{G}$, service provider g, such that $\hat{c}_g = \min_{i \in \mathcal{G}}[\hat{c}_i]$, is selected and is added to $\mathcal{K}$. Its paired cost is $\hat{c}_g^p = \min_{i \in \mathcal{G} \setminus \{g\}}[\hat{c}_i]$. Then, as in the Paired mechanism, the consumer computes the optimal procurement strategy using only providers in $\mathcal{K}$ and their paired costs, and the transfers are defined similarly. While in the Halving mechanism the size of the set of candidate providers is smaller than that of the Pairing mechanism ($|\mathcal{K}| = \lfloor m/4 \rfloor + 1$ as opposed to $|\mathcal{K}| = \lfloor m/2 \rfloor$), using the larger set $\mathcal{G}$ increases the likelihood that a provider with a low paired cost will be placed in $\mathcal{K}$.

Theorem 2. *The Pairing and Halving mechanisms are incentive compatible in dominant strategies and post-execution individually rational.*

Proof. Since the pairs and $\mathcal{G}$ are formed independently of the providers' reports, the proof follows from Theorem 1. □

We note that there are many possible variations of these mechanisms since different ways to group providers could be used. All the mechanisms, however, share some key features. First, the size of $\mathcal{K}$ is solely determined by the *number of providers* and thus does not rely on the consumer choosing an appropriate value. Second, the mechanisms do not require *a priori* information about the costs of the providers, since all transfers are determined by the costs of providers who are not members of $\mathcal{K}$. Finally, they implement *discriminatory pricing* (i.e., different providers receive different payments), information which is then used to form the optimal procurement strategy (given $\mathcal{K}$). In order to ascertain whether these variations offer any real benefit, compared to the uniform pricing mechanism, in practice, we experimentally evaluate them in Section 6.2.

5.3. Unknown costs and duration probabilities

In this section, we relax the assumption that the duration distributions are known, and consider mechanisms which need to elicit both the distributions as well as the costs. To this end, we first show that the marginal contribution mechanism, an example of a VCG mechanism, from Section 5.2.1 no longer exhibits our desired properties and, in particular, providers

[19] Simulations, for example, or knowledge based on past experience may be ways to help determine k.

have an incentive to misreport their duration distributions. We then introduce a modified mechanism, which we refer to as the *Execution-Contingent VCG* mechanism, where the transfers are contingent on the actual execution of the schedule and on whether or not the task succeeded.

5.3.1. Failure of the standard VCG mechanism

Consider the marginal contribution mechanism, introduced in Section 5.2.1, with the modification that each provider, i, is asked to report both its cost, $\hat{c}_i$, and its duration probability, $\hat{F}_i$. The transfers for this mechanism are calculated as follows:

$$\tau_i = \overline{w}_{-i}\big(\rho^*(\hat{c}, \hat{F})\big) - \overline{w}\big(\rho^*(\hat{c}_{-i}, \hat{F}_{-i})\big)$$

As in the previous section where providers only reported their costs, provider i has no influence on the second term of the transfer function, $\overline{w}(\rho^*(\hat{c}_{-i}, \hat{F}_{-i}))$, since this is the expected social welfare that would have been achieved if provider i had not participated in the mechanism in the first place. However, we now show, by example, that a provider can improve its transfer by misreporting F_i in such a way that the first term of the transfer is increased, thus resulting in higher expected utility for the provider.

Example 2. For the sake of simplicity, suppose that provider i only misreports its duration distribution and that all other providers report truthfully. Also suppose that $\rho^*(c, F) = \langle (i, 0) \rangle$. That is, given the true F_i, the optimal schedule is to only invoke provider i and to do so without delay. Now, suppose there exists an alternative distribution, F'_i, such that $\rho^*(c, (F'_i, F_{-i})) = \rho^*(c, F) = \langle (i, 0) \rangle$ (i.e., the schedule remains unchanged) and $F'_i(D) > F_i(D)$ (i.e., the probability of success by the deadline is higher).[20] Clearly, since an increasing probability of success increases the consumer's utility, $\overline{w}(\rho^*(c, (F'_i, F_{-i}))) > \overline{w}(\rho^*(c, F))$, but also $\overline{w}_{-i}(\rho^*(c, (F'_i, F_{-i}))) > \overline{w}_{-i}(\rho^*(c, F))$, resulting in an increase of transfers τ_i when reporting F'_i instead of F_i. Since reporting F'_i has no impact on the probability of being invoked (i.e., the allocation remains unchanged), provider i is better off doing so.

In the above example, the providers have an incentive to misreport their distributions as this will increase the *perceived* expected utility of other agents in the system, and thereby increase the perceived expected social welfare. This, in turn, leads to an increase in the transfers. In this particular case, the provider was able to increase the perceived expected utility of the consumer agent by increasing the probability of success. However, it is equally possible to construct examples that increase the expected utility of other providers.

Technically, the marginal contribution mechanism fails here because the expected utility of an agent (either the consumer or one of the providers) depends not only on the schedule, but also on the private information of other agents in the system (i.e., in this case the information about the duration distributions). Such settings are known as settings with *interdependent types* [28], and it has been shown that, in general, in situations where agents have interdependent types, it is impossible to design a mechanism which ensures that the chosen outcome maximises social welfare and is incentive compatible in dominant strategies (see, for example, Jehiel and Moldovanu [26]). Therefore, we need to make a concession on one of these properties and so, to this end, we introduce a mechanism that still maximises social welfare, but in which the information revealed by the service providers may depend on the actions taken by others. We make this particular choice, because we are still interested in using redundancy optimally to complete the task despite the execution uncertainty and because it turns out that we can obtain a slightly weaker notion of incentive compatibility, where truthtelling is a Nash equilibrium for all participants, regardless of their particular types and without the need for prior distributions over these.

5.3.2. Execution-Contingent VCG

In this section, we introduce a modification of our marginal contribution mechanism, where the transfers made to the service providers are *contingent* on the outcome of the execution of the procurement strategy. We show that this modification results in a mechanism which is able to elicit both the costs and the duration distributions from the service providers.

As before, each service provider, i, is asked to report its cost, $\hat{c}_i$, and its duration distribution, $\hat{F}_i$. Using the reported costs and duration functions, the consumer finds the optimal procurement strategy $\rho^*(\hat{c}, \hat{F}) = \operatorname{argmax}_{\rho \in \mathcal{P}} \rho(\hat{c}, \hat{F})$. This strategy is then executed, and *upon completion of execution* and once the outcome is known (i.e., whether or not the task was completed successfully before the deadline), the transfers of the providers are determined. Recall, from Section 3 that X_ρ denotes the *actual completion time* of the task, and $\mathcal{I}_\rho$ denotes the *set of invoked providers*. Then

$$\tau_i = \begin{cases} V - \sum_{j \in \mathcal{I}_{\rho^*(\hat{c}, \hat{F})} \setminus \{i\}} c_j - \overline{w}(\rho^*(\hat{c}_{-i}, \hat{F}_{-i})) & \text{if } X_{\rho^*(\hat{c}, \hat{F})} \leqslant D \\ - \sum_{j \in \mathcal{I}_{\rho^*(\hat{c}, \hat{F})} \setminus \{i\}} c_j - \overline{w}(\rho^*(\hat{c}_{-i}, \hat{F}_{-i})) & \text{otherwise} \end{cases} \tag{35}$$

Now, as we will show in the remainder of this section, the Execution-Contingent VCG mechanism has a number of desirable properties, but these properties hold under slightly weaker solution concepts compared to the mechanisms described

[20] As a concrete example, assume that $F_i(D) = 0.9$ and $F'_i(D) = 0.99$, and that $F_i(x) = F'_i(x) = 0$, for all $x < D$.

in Section 5.2 for known duration distributions. In particular, it is not always desirable for a provider to reveal its own cost and duration distribution truthfully *if others are not truthful*. To illustrate this, we describe a simple example:

Example 3. As in Example 1, assume that $V = 10$ and there are two providers with $c_1 = c_2 = 5$. Again, there are two time steps of interest before the deadline, $t_1 = 0$ and t_2, but we now assume that provider 1 can never complete the task in time, i.e., $F_1(D) = F_1(D - t_2) = 0$, while provider 2 remains unchanged, i.e., $F_2(D) = F_2(D - t_2) = 0.8$. Assume that provider 1 reports its cost truthfully, but lies about its duration distribution, reporting $\hat{F}_1(D) = \hat{F}_1(D - t_2) = 0.9$. Given this, if provider 2 truthfully reports its cost and duration distribution, the mechanism will select $\rho^*(\hat{c}, \hat{F}) = \langle (1, 0), (2, t_2) \rangle$. In this case, the expected utility of provider 2 after learning the allocation, and assuming that it knows the true distribution F_1, is as follows:

$$\begin{aligned}\overline{u}_2\big(\rho^*\big(\langle \hat{c}_1, c_2\rangle, \langle \hat{F}_1, F_2\rangle\big)\big|F_1\big) &= F_2(D - t_2) \cdot V - \hat{c}_1 - c_2 - \overline{w}\big(\rho^*(\hat{c}_1, \hat{F}_1)\big) \\ &= 0.8 \cdot 10 - 5 - 5 - 4 \\ &= -6\end{aligned}$$

Clearly, provider 2 now has a negative expected utility and therefore an incentive to change the allocation so that it is no longer included, thus deriving a utility of zero. It could do this, for example, by reporting $\hat{c}_2 = 10$ or $\hat{F}_2(D) = \hat{F}_2(D - t_2) = 0$.

Since the mechanism is not incentive compatible in dominant strategies, as we just illustrated in the example, we will, instead, try to achieve a weaker notion of incentive compatibility.

Definition 8 *(Ex-post incentive compatibility).* A mechanism is *ex-post incentive compatible*, if, for each provider i with c_i and F_i, and for all possible cost functions and duration distributions of other providers, c_{-i} and F_{-i}, and for all $\hat{c}_i \neq c_i$ and $\hat{F}_i \neq F_i$,

$$\overline{u}_i\big(\rho^*\big(\langle c_i, c_{-i}\rangle, \langle F_i, F_{-i}\rangle\big)\big) \geqslant \overline{u}_i\big(\rho^*\big(\langle \hat{c}_i, c_{-i}\rangle, \langle \hat{F}_i, F_{-i}\rangle\big)\big)$$

In words, a mechanism is ex-post incentive compatible, if, when all service providers but i report their cost and duration distributions truthfully, then no matter what this revealed information is, provider i maximises its expected utility by truthfully reporting its own cost and duration distributions. This is a weaker notion of incentive compatibility than incentive compatibility in dominant strategies, since truthtelling by provider i relies on all other providers also reporting their information truthfully. However, it is stronger than Bayesian incentive compatibility, because it does not depend on prior knowledge of the other providers' private information and because truthtelling is a Nash equilibrium, even when types are revealed after the allocation. Hence, it is often regarded as a realistic solution concept in the mechanism design literature (see, for example, Bergemann and Morris [4] for a detailed discussion).

Theorem 3. *The Execution-Contingent VCG mechanism is*:

- *Ex-post incentive compatible, and*
- *Individually rational.*

Proof. Assume that all service providers but i truthfully report their costs and duration distributions. That is, they report c_{-i} and F_{-i}. Then, if provider i reports $\hat{c}_i$ and $\hat{F}_i$, its expected utility is

$$\begin{aligned}\overline{u}_i\big(\rho^*\big(\langle \hat{c}_i, c_{-i}\rangle, \langle \hat{F}_i, F_{-i}\rangle\big)\big) &= \overline{w}_{-i}\big(\rho^*\big(\langle \hat{c}_i, c_{-i}\rangle, \langle \hat{F}_i, F_{-i}\rangle\big)\big|c_{-i}, F\big) - \overline{w}\big(\rho^*(c_{-i}, F_{-i})\big) \\ &\quad - c_i \cdot \big(1 - \mathrm{Prob}(X_{\rho^*(\langle \hat{c}_i, c_{-i}\rangle, \langle \hat{F}_i, F_{-i}\rangle)} \leqslant t_i)\big) \\ &= \overline{w}\big(\rho^*\big(\langle \hat{c}_i, c_{-i}\rangle, \langle \hat{F}_i, F_{-i}\rangle\big)\big|c, F\big) - \overline{w}\big(\rho^*(c_{-i}, F_{-i})\big) \qquad (36)\end{aligned}$$

First, we note that the second term on the RHS is independent of provider i's reported cost and duration distribution. Thus, there is nothing that provider i can do to change this value, given the reports of the other providers. Secondly, the first term of the RHS is computed *after the execution of procurement strategy* ρ^*. While the selection of ρ^* depends on the reported cost and duration probabilities, the actual outcome upon execution depends on the true distribution durations. As a result, note that:

$$\overline{w}\big(\rho^*\big(\langle c_i, c_{-i}\rangle, \langle F_i, F_{-i}\rangle\big)\big|c, F\big) \geqslant \overline{w}\big(\rho^*\big(\langle \hat{c}_i, c_{-i}\rangle, \langle \hat{F}_i, F_{-i}\rangle\big)\big|c, F\big)$$

by definition of ρ^*. Thus, if all other providers truthfully report their costs and duration distributions, provider i is also best off revealing its information truthfully, since this will result in the mechanism selecting the procurement strategy which optimises the social welfare in expectation, which leads to the expected utility maximisation of provider i.

The Execution-Contingent VCG mechanism is also individually rational, since

$$\overline{w}\big(\rho^*(c, F)\big) \geqslant \overline{w}\big(\rho^*(c_{-i}, F_{-i})\big)$$

implying that $\bar{u}_i(\rho^*(c, F)) \geqslant 0$. □

In Section 5.2.1, we identified two main drawbacks of the marginal contribution mechanism: the computational requirements to calculate the optimal schedule, and the fact that service providers may have an incentive to de-commit. Interestingly, the latter is no longer a problem when using Execution-Contingent VCG, despite the possibility that the post-execution utility of the provider may become negative. This is because the utility is calculated based on what actually happened, and any increase in the post-execution social welfare results in the same increase in transfers. Therefore, there is no need to impose additional penalties or a deposit to enforce the schedule. The first issue relating to the computational overheard of the marginal contribution mechanism still arises with the Execution-Contingent VCG. To address this problem, in the next part, we investigate how we can approximate the solution to the optimal schedule, while maintaining the properties of the mechanism. Contrary to Section 5.2, however, we do so using the same mechanism and instead restrict the set of possible outcomes.

5.3.3. Approximating the optimal procurement strategy

As mentioned in Section 4.3.2, computing the optimal procurement strategy becomes intractable as the number of available providers increases, and this is of particular importance in our domain, as the consumer has limited computational resources. However, as discussed in Section 5.2.1, replacing the optimal procurement strategy with a sub-optimal one obtained through use of a heuristic or approximation algorithm, can destroy the incentive properties of the underlying mechanism [36]. In Section 5.2, we chose to address the computational problem by considering alternative, simpler, mechanisms which required less computation on the part of the consumer. However, these mechanisms are not directly applicable in the current setting, because computing the optimal procurement strategy is an intrinsic part of the mechanism. For this reason, we now propose an alternative approach for reducing the computational burden on the consumer.

Nisan and Ronen showed that it is sometimes possible to have mechanisms which knowingly used sub-optimal outcomes [36]. They proposed that instead of changing the algorithm for finding the optimal outcome (in our case, the optimal procurement strategy), one could restrict the *set of possible outcomes*, and then run the optimal algorithm on this restricted set.[21]

In what follows, we apply this approach to our procurement problem, and show that the Execution-Contingent VCG mechanism is incentive compatible for appropriately restricted outcome spaces. Furthermore, we show that this approximation admits a *polynomial-time* algorithm to calculate the optimal (within the space of allowable outcomes) procurement strategy. We will also show, however, that the approximation can (in the worst case) be arbitrarily far from the optimal outcome. Nevertheless, in practice we find that the outcomes are often close to optimal, as we will show in Section 6 where we analyse the approximation empirically.

We start by showing that the EC-VCG mechanism is still incentive compatible when certain approximation schemes are used. For this, we let η denote the maximum number of service providers that can be selected as part of a procurement strategy, and let $\mathcal{P}_\eta = \{\rho \in \mathcal{P}\colon |\rho| \leqslant \eta)$ represent the *reduced set* of strategies. When $\eta = 1$, then the reduced set of strategies contains only procurement strategies with no redundancy, while if $\eta = m$ then $\mathcal{P}_\eta$ is the full procurement strategy space. We propose applying the Execution-Contingent VCG mechanism, but selecting only procurement strategies from the set $\mathcal{P}_\eta$. We can show that this restricted version of the Execution-Contingent VCG mechanism is incentive compatible and individually rational under the assumption that η is using no information about the service providers (i.e. before the mechanism starts).

Theorem 4. *For any* $1 \leqslant \eta \leqslant m$*, if the allocation is given by* $\operatorname{argmax}_{\rho\in\mathcal{P}_\eta} \overline{w}(\rho|\hat{c}, \hat{F})$*, the Execution-Contingent VCG mechanism with the reduced set of procurement strategies,* $\mathcal{P}_\eta$*, is incentive compatible and individually rational.*

Proof. Since η is independent of any of the providers' reports, no provider can increase the social welfare, and hence its transfers, by misreporting. Therefore, the proof follows directly from Theorem 3. □

Given this, we now show that, once the parameter η is set, then finding the optimal procurement strategy in $\mathcal{P}_\eta$ becomes polynomial in the number of possible service providers, m. We illustrate this by considering two different scenarios. First, we consider the situation where the optimal invocation times for providers can be found analytically, given a set of providers and their ordering in the procurement strategy (such settings were discussed in Sections 4.2.1 and 4.2.2 for independent and correlated duration distributions, respectively). The problem of finding the optimal procurement strategy then reduces to the problem of finding the optimal ordering of the providers, among all sets of η providers. This is equivalent to searching through all possible ordered subsets of set M of size η, which has size $m!/(m-\eta)!$. Once the optimal procurement schedule

[21] In particular, if an algorithm is *maximal-in-range*, then VCG-based mechanisms, applied to the restricted problem, are incentive compatible. See [36] for details.

is found, then the transfers for all providers must be computed. If a provider is not in the optimal procurement strategy then their transfer is automatically set to zero, and thus we only need to explicitly compute the transfers for at most η providers in the optimal procurement strategy. The overall cost of this is $\eta \cdot (m-1)!/(m-\eta)!$, which results in the overall complexity of $O(m^\eta)$ for running the mechanism.

In the case that finding the optimal invocation schedule does not allow for a closed-form analytical solution, another approach is to *discretise time*. Let T denote the total number of discrete time slots before the deadline D.[22] Now, given an ordered set of candidate providers, each of these providers has at most T possible invocation times, except for the first provider who should be invoked immediately (recall from Section 4 that it is always optimal to invoke the first provider with no delay). Since there are at most η candidate providers, finding the optimal invocation times therefore requires searching through less than $T^{\eta-1}$ combinations. Together with finding the optimal set of ordered candidate providers, this results in a time complexity of $O(m^\eta \cdot T^{\eta-1})$.

While these approximations have a desirable computational complexity, they may result in sub-optimal solutions. This is because they restrict the set of solutions and this can yield an outcome that is significantly worse than the optimal — especially when the optimal strategy, ρ^*, contains many more than η providers. In fact, in the following theorem, we show that it can be arbitrarily far from the optimal.

Theorem 5. *For any $\eta \geqslant 1$, there exists an m, F and c, such that the ratio between the expected social welfare of the optimal solution and the approximate solution is at least b, for* any *$b \geqslant 1$. That is*:

$$\frac{\overline{w}(\rho^*(c,F))}{\operatorname{argmax}_{\rho\in\mathcal{P}_\eta}\overline{w}(\rho|c,F)} \geqslant b \tag{37}$$

Proof. We prove this by showing how to choose m, F and c, so that the above holds. For simplicity, we assume here that durations of different providers are independent, as shown in Eq. (10). For all i, we let $c_i = 0$ and $F_i(D) = 1 - f$, with $1 > f \geqslant (1-\frac{1}{b})^{\frac{1}{\eta}}$. Clearly, as providers are free in this example, it is always optimal to invoke all available providers at time $t = 0$. Given this, we can now choose m, so that Eq. (37) holds:

$$\begin{aligned} b &\leqslant \frac{\overline{w}(\rho^*(c,F))}{\operatorname{argmax}_{\rho\in\mathcal{P}_\eta}\overline{w}(\rho|\hat{c},\hat{F})} \\ \Leftrightarrow \quad b &\leqslant \frac{(1-f^m)\cdot V}{(1-f^\eta)\cdot V} \\ \Leftrightarrow \quad m &\geqslant \frac{\ln(1-b\cdot(1-f^\eta))}{\ln(f)} \end{aligned} \tag{38}$$

Due to our initial constraints for f, this can always be satisfied. □

To conclude, we have shown that, through limiting the number of possible outcomes, we can obtain a solution which is polynomial in the number of service providers, while maintaining the desired properties of the mechanism. However, we have also demonstrated that, in theory, this approximation can be arbitrarily far from the optimal. Nevertheless, we believe that, in realistic settings, where providers are generally costly, the benefit of increased redundancy diminishes with the number of providers. Therefore, an approximate solution may often be close to the optimal, even when η is chosen to be significantly less than m. To this end, in the next section, we empirically evaluate all of the mechanisms discussed thus far, including the approximation.

6. Empirical evaluation

Having described optimal procurement strategies for scenarios with full information, as well as a number of mechanisms that incentivise providers to reveal their capabilities truthfully, we now evaluate these approaches in a variety of simulated environments. The purpose of this evaluation is two-fold and it should be seen as a complement to the theoretical results of the previous sections. First, we examine the potential benefit of procuring multiple providers and compare this to current approaches that invoke only single providers or use simple heuristics (Section 6.1). While our example from Section 4.1 showed that redundancy can be beneficial, we use experiments here to quantify these benefits over a much wider range of realistic environments. Second, we consider the mechanisms proposed in Section 5 and investigate to what extent their use entails a loss in the consumer's utility and in overall efficiency (Section 6.2). Again, while our results proved a range of desirable properties, some of our mechanisms are inherently suboptimal, due to the use of heuristics or approximations, and they also generally require an additional investment by the consumer to incentivise truthfulness. In this context, our

[22] Note that these time slots are not required to be equally spaced. However, it is important that they are set *independent of the reports* of the providers. Otherwise, the incentive properties may no longer hold.

experiments help us quantify these losses in utility over a range of settings. Finally, in Section 6.3, we discuss how our theoretical and empirical results can be combined to help the consumer choose the right mechanism for a particular setting.

6.1. Full information

In this section, we start by considering the full information case, where the consumer has full knowledge of the providers' costs (c_i) and duration distributions (F_i). We also assume that providers charge their true costs, i.e., $\tau_i = c_i$, and therefore the social welfare is, in this case, equal to the consumers's expected utility (as described in Section 4). Throughout the section, we compare the average expected social welfare ($\overline{w}(\rho)$) obtained by the optimal procurement strategy[23] described in Section 4 (*Optimal*) to two benchmark strategies that are designed to represent current approaches:

- *Single*: This strategy selects the single provider that individually maximises the consumer's expected utility. As such, it represents approaches that do not use redundancy at all (in fact, it is the *optimal* strategy given this restriction).
- *Timeout*(p, t): This strategy first orders all providers using a preference ordering given by parameter $p \in \{\texttt{cost}, \texttt{time}, \texttt{balanced}\}$. These choices for p, respectively, order all providers by ascending cost, c_i, mean duration, $1/\lambda_i$, or a combination, c_i/λ_i. The strategy then invokes the providers in that order, leaving a waiting time of t between successive invocations. This continues until either the task is completed or the deadline is reached. The *Timeout* strategy represents approaches that are often used in existing applications to deal with uncertainty (see Section 2.1).

Comparing our Optimal strategy to these benchmarks allows us to quantify the benefit of using redundancy in an optimal and principled manner to deal with execution uncertainty. Here, our evaluation is guided by the following hypothesis:

Hypothesis 1. Compared to current procurement approaches, using redundancy results in a significant increase in utility over a wide range of environments.

We divide this section into two parts, corresponding to the two models of uncertainty we have covered in detail in this paper — first, we test our approach in environments where service durations are independently distributed (Section 6.1.1) and then we consider those environments where service durations are perfectly correlated (Section 6.1.2).

6.1.1. Independent durations

To test Hypothesis 1 in settings with independent durations, we randomly generate each provider i by drawing its cost c_i and duration rate λ_i independently and uniformly at random from $[0, 1]$. To consider a range of settings, tasks have either a low ($V_{\text{low}} = 2$) or a high value ($V_{\text{high}} = 8$) and their deadline is either normal ($D_{\text{normal}} = 2$) or urgent ($D_{\text{urgent}} = 0.5$). Furthermore, throughout our evaluation we repeat all experiments 1000 times and use ANOVA and t-tests to ensure statistical significance at the $p < 0.05$ level. As the associated confidence intervals are generally small, we omit these in most figures for clarity.

The results are shown in Fig. 6. Here, we vary the number of providers in the system and plot the average expected social welfare as a proportion of V. Observing these trends, it is obvious that using redundancy can significantly improve the consumer's utility and does so in almost all settings considered. In fact, when averaging over all cases considered, the *Optimal* approach achieves more than a 40% improvement over the *Single* approach. In general, this gain becomes more pronounced when the number of providers in the system increases, as there are more candidates to invoke redundantly and the consumer can also be more selective about which providers to procure. Similarly, the gain achieved through redundancy increases as the deadline becomes shorter. This is because redundancy allows the consumer to achieve a high success probability, while the *Single* approach is limited by the single most reliable provider available. Finally, we note that the gain in utility rises for higher task values, as this justifies the higher investment caused by using redundancy.

Hence, in the results shown here, the most marked improvement over the *Single* strategy is obtained when the deadline is short (D_{urgent}), the task value is high (V_{high}) and there are many potential providers ($m = 50$). Here, the *Single* approach achieves 35.82% of V, while the *Optimal* achieves 82.68% — a 130% improvement that is obtained by using procurement strategies with an average of 10–11 providers (7–8 of which are typically invoked).

For comparison, the graphs also show two representative *Timeout* strategies — here, we chose $p = \texttt{balanced}$ with $t = 0.04$ and $t = 0.53$, because these represent the parameter choices that obtain the best average utility for the settings with $D_{\text{urgent}}, V_{\text{high}}$ and $D_{\text{normal}}, V_{\text{low}}$, respectively.[24] It is obvious that these heuristic strategies do not consistently achieve good results. The *Timeout*(*balanced*, 0.53) strategy does perform well in both settings with long deadlines, because it balances the cost of providers with their speed and also benefits from some redundancy, but it performs poorly in settings with short deadlines. In fact, it performs worse here than the *Single* approach, because it does not reason about the probability of completing the task within the tight deadline and therefore chooses providers that do not even have a significant probability

[23] This is found using our branch-and-bound algorithm when there are up to twelve providers. We then use the heuristic algorithm to obtain a lower bound for the optimal when there are more providers. However, as we show later, the heuristic obtains near-optimal results.

[24] We obtained these by discretising t in 0.01 steps and then testing all possible parameter combinations.

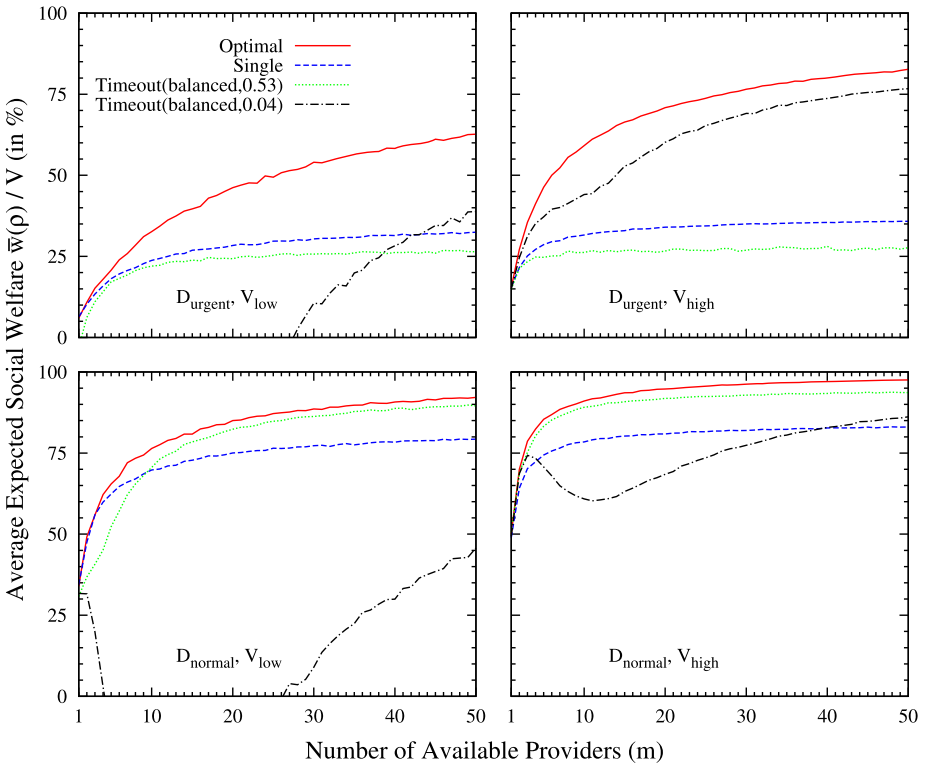

Fig. 6. Performance in full information setting.

of success. Similarly, while the *Timeout*(*balanced*, 0.04) performs well in the scenario with a short deadline and high value, it leads to extremely poor or erratic performance in other settings. In fact, here, the strategy often invokes far more providers than necessary and therefore incurs a large negative utility (which is outside the range of the graphs). In summary, the *Timeout* strategy only performs well if its parameters are carefully tuned for its particular environment, and even then, it is outperformed by our *Optimal* strategy. Overall, therefore, we can conclude that Hypothesis 1 holds in settings with independent service durations.

Additionally, we note that when solving the above problems, our branch-and-bound approach significantly reduces the computational time required when compared to a brute-force algorithm. For example, when there are 12 providers and we consider V_{high} and D_{urgent}, a brute force approach searches over 1.3 billion provider orderings, which takes an average 3.3 hours (using a Java implementation on a Windows-based Intel 2.2 GHz laptop with 4 GB RAM). By contrast, the branch-and-bound algorithm searches an average of 69 200 orderings (0.005% of the total search space), finding the optimal in just over two seconds. While the latter still finds solutions for 18–23 providers in minutes (where the brute-force would take over 2×10^{10} years — longer than the age of the universe), our heuristic approach is better suited for larger settings with hundreds or thousands of providers. To investigate how its performance compares to the optimal, we have applied both to all settings described above with twelve or fewer providers. Here, we found no statistically significant difference to the optimal strategy (the graphs are not shown here for reasons of space).

In the following, we investigate environments with perfectly correlated durations.

6.1.2. Correlated durations

To test Hypothesis 1 in these settings, we repeat the same experiments as above, but now assume that durations are perfectly correlated. Somewhat surprisingly, the *Optimal* approach here performs identically to the *Single* approach, i.e., it also always procures only the service provider that individually yields the highest expected utility. Using redundancy, therefore, does not result in a higher utility for the consumer here, thus contradicting Hypothesis 1 in environments with perfectly correlated durations. As we will explore in more detail in the remainder of this section, this is here due to the relatively short deadline that we considered in our previous experiments. More specifically, as a result of the complete correlation, any additional providers need some time to catch up with previously invoked providers and this negates the advantage of procuring multiple providers when the deadline is short. Rather, it is best to immediately procure a better provider, in order to achieve a high probability of success. This is in contrast to the independent case, where any additional provider immediately contributes positively to the overall success probability. Overall, this is an interesting result, indicating that the simple greedy approach represented by the *Single* strategy is adequate and often optimal when durations are perfectly correlated.

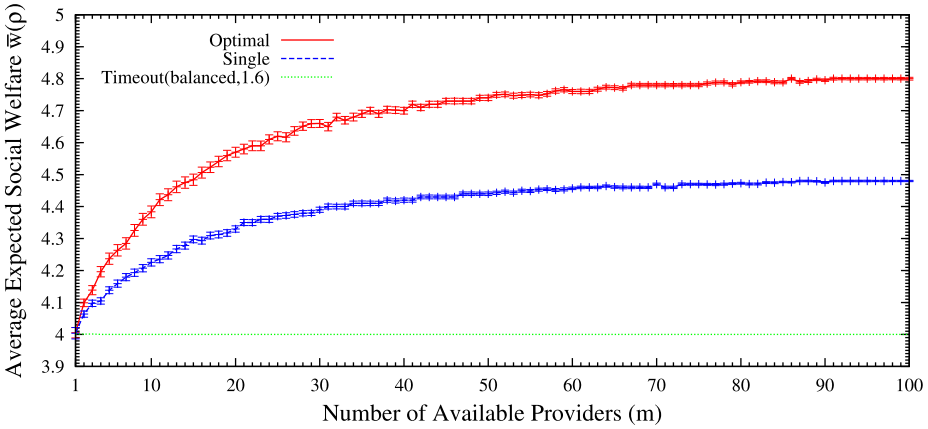

Fig. 7. Performance with perfectly correlated durations.

However, we note here that there are significant exceptions to this — one of which we have already highlighted in Section 4.1. In that example, using redundancy resulted in a considerable 50% improvement over the *Single* approach. To investigate in more detail the types of environments where such improvements may be found in practice, we note the following features in our example which set it apart from the settings we have considered so far in our evaluation:

- **Long deadline:** The task in the example had a long deadline relative to the speed of at least one provider. This made it feasible to delay the invocation of the second provider without causing a significant impact on its success probability. This is in contrast to the experiments carried out here, where even the fastest providers (i.e., when $\lambda_i = 1$) have a non-negligible failure probability, which is quickly decreased by any further delays. Furthermore, any redundantly invoked providers typically need some time to catch up with previously invoked providers, i.e., to reach the time t where the capacity of the redundantly invoked provider i is equal to that of its predecessor. Again, this reduces the potential benefit of using redundancy in settings with short deadlines.
- **Correlated cost and quality:** The providers in the example displayed a correlation between their quality and cost — one was cheap and slow, while the other one was expensive and fast. This required making an explicit trade-off between these, and the redundant approach was able to benefit from combining both providers. In the setting considered above, we assumed costs and qualities were independent, thus often leading to the situation where a particular provider clearly dominated the others.

With this in mind, we now consider a variation of our experimental setup. In particular, we use deadline $D_{\text{normal}} = 2$ and a high value $V_{\text{high}} = 8$, but we now draw λ_i uniformly at random from the interval $[0, 30]$ and determine the cost as $c_i = 4 \cdot (1 - e^{-\lambda_i})$. This means that the deadline is now very long relative to some of the providers and we also correlate the quality with the cost, which is linearly dependent on the success probability within a unit time step. As the branch-and-bound algorithm is significantly faster in the correlated setting, we solve the problem optimally for up to 100 providers here (this improvement in speed is due to the fact that we can immediately select an optimal ordering, as described in Section 4.3.1).

The results for this modified setting are shown in Fig. 7. Not surprisingly, the improvement is still not as large as in environments with independent durations, because the overall success probability still depends only on the single best provider (whereas in independent settings, every provider contributes to increasing this success probability). Nevertheless, we can now see that the *Optimal* strategy here still offers a significant advantage over the *Single* strategy. In fact, over the environments tested here, the average improvement is 6.12%. Furthermore, this relative improvement increases with the number of available providers, resulting in a 7.14% improvement when there are 100 providers. We note that the *Timeout* strategy performs poorly here. Despite choosing the best p and t parameters, as before, the strategy is inherently ineffective for the correlated case, because it invokes similar providers in sequence, which often does not increase the success probability at all and instead incurs additional costs.

These results indicate that using redundancy can be beneficial even in perfectly correlated settings, although the improvement is not as pronounced as in the independent case and there are some environments where redundancy offers no additional benefit over the *Single* strategy. However, we have so far assumed that the task difficulty follows an exponential distribution. We believe that this may be inherently less suitable to redundant procurement when durations are correlated, as it has a constant hazard rate and a density that quickly diminishes over time. Thus, a single reasonably fast provider is often the best choice for the consumer. In the following, we consider a different scenario, where redundancy may offer a more substantial benefit.

This new scenario is based on the observation that many realistic computational problems exhibit phase transitions in their difficulty [7]. That is, although the problem may be hard to solve in the general case, many instances can be solved

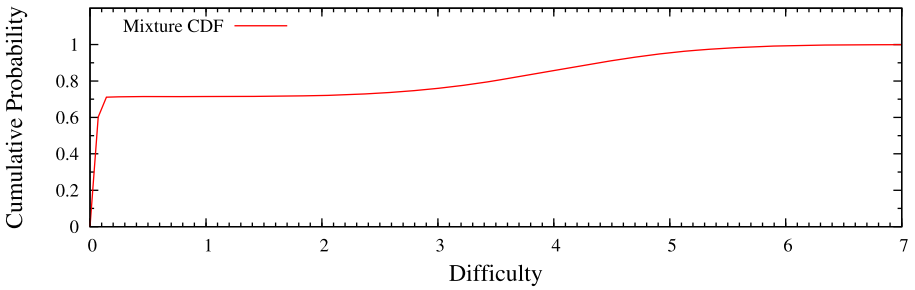

Fig. 8. Bi-modal distribution to simulate distinct levels of difficulty.

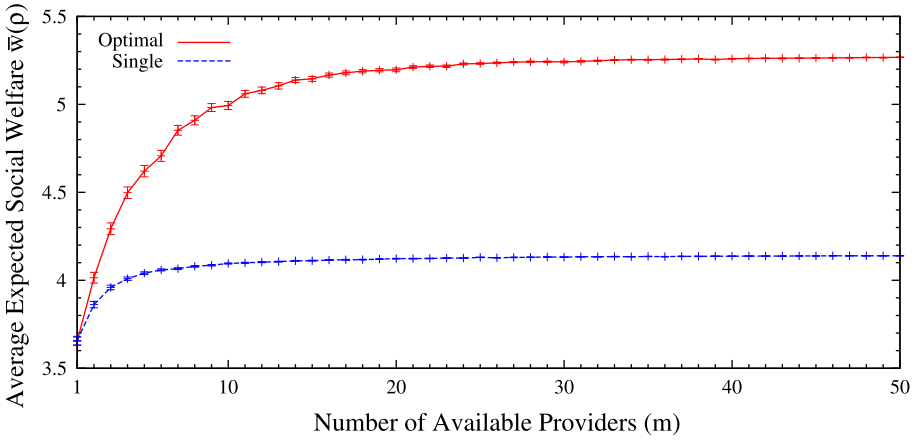

Fig. 9. Performance with perfectly correlated durations and bi-modal distribution.

quickly in polynomial time. Yet, when the problem parameters exhibit certain properties, the run-time becomes exponential. This typically happens abruptly and can be difficult to predict a priori for individual instances.

As we believe that such a setting often occurs when there is uncertainty about the task difficulty (i.e., when provider durations are correlated), we extend our evaluation here to settings where the task difficulty is a multi-modal distribution. More specifically, we assume that the task difficulty falls either into the class of relatively easy problems or into the class of hard problems, which have a significantly higher and more variable run-time. Although the difficulty is uncertain *a priori*, the consumer may still have a probability distribution, based on past problems and conditional on certain features of the input data. To model this in our evaluation, we assume that the task difficulty follows a mixture of two normal distributions that are truncated at 0, as shown in Fig. 8.

More specifically, in this setting, we again assume V_{high} and D_{normal}. The problem difficulty Y is given by the density function $g(y) = \frac{5}{7}\frac{n_{0,0.5}(y)}{1-N_{0,0.5}(0)} + \frac{2}{7}\frac{n_{4,1}(y)}{1-N_{4,1}(0)}$, where $n_{\lambda,\sigma}(y)$ is the density function of a normal distribution with mean λ and standard deviation σ and $N_{\lambda,\sigma}(y)$ the corresponding cumulative distribution function. The completion time of a particular provider i is given by $q_i(t) = \lambda_i t$, where λ_i is drawn uniformly at random from $[0, 2]$. As before, we correlate the quality and cost by setting $c_i = 4 \cdot (1 - \mathrm{e}^{-\lambda_i})$. It should be noted that F_i here no longer follows an exponential distribution, as we assumed in Section 4.2.2. For this reason, we restrict our invocation times to ten uniformly spaced time steps, $(0, \frac{D}{10}, \frac{2D}{10}, \ldots, \frac{9D}{10})$, and then use a brute-force search to find the optimal schedule for a given ordering of providers.

The results are shown in Fig. 9 and indicate that the *Optimal* approach offers a more significant improvement over the *Single* approach when the problem difficulty follows a multi-modal distribution (we omit the *Timeout* strategy, as it again performs very poorly). In fact, over the cases tested here, the average improvement is 24.1% and in some cases reaches 27.3% (when $m = 50$). Compared to the exponential distribution, these problems are more conducive to using redundancy, because different providers can be used to specifically target parts of the distribution. For example, initially, a cheap provider can be procured to cover problems with a low run-time (in this example, this occurs in over 70% of all cases); then, when a certain threshold is reached, a fast provider is procured, because the task at hand is most likely one that is difficult to solve.

In conclusion, we have so far shown that redundancy offers a significant benefit over the procurement of single providers. This is particularly the case when duration distributions are independent, because each additional provider increases the probability of success and many cheap providers can be combined to obtain a high quality of service. We showed in general that Hypothesis 1 holds in these environments.

When durations are perfectly correlated, this improvement is generally lower, because the consumer's success probability depends only on the capacity of the best service provider and because even faster providers need some time to catch up

with previously invoked providers. In some environments, especially with short deadlines, this means that redundancy may offer no benefit over the *Single* approach. However, when deadlines are long, costs and service qualities are correlated, or when the uncertainty follows a multi-modal distribution, redundancy can offer a clear advantage over the *Single* approach.

While we have so far considered settings where the consumer has full information about the providers' capabilities and costs, we now move on to settings where this information is private.

6.2. Private information

Now we consider the mechanisms described in Section 5. As mentioned there, these mechanisms have a number of desirable properties (including incentive compatibility and individual rationality), but some of them do not result in a welfare-maximising outcome and generally require the centre to pay providers more than their costs. However, our theoretical results do not quantify these losses in realistic settings. To address this shortcoming, we measure the performance of our mechanisms empirically here, by applying them to a wide range of settings. To this end, we are interested in the loss of utility in terms of social welfare (i.e., how far the solutions obtained by the mechanisms are from the optimal solution), as well as in the inherent loss of utility to the consumer caused by its lack of information about the providers' costs and duration distributions (this latter cost is also called the *information rent* in the mechanism design literature [34, Ch. 23]).

We start in Section 6.2.1 by considering the uniform and discriminatory pricing mechanisms we proposed to deal with scenarios where the duration probabilities are known by the consumer and only the costs are private information. We omit a specific discussion of our first mechanism, the marginal contribution mechanism, because its performance in terms of expected social welfare and the consumer's expected utility is equal to the Execution-Contingent VCG, which we cover in detail in Section 6.2.2. As discussed in Sections 5.2.2 and 5.2.3, both the uniform and the discriminatory pricing mechanisms for these scenarios are generally sub-optimal, but we expect some to perform better in certain settings. Hence, in addition to quantifying the overall utility obtained by the mechanisms, our investigation is driven by the following hypotheses:

Hypothesis 2. When we have sufficient information to choose k optimally, then the uniform pricing mechanism consistently outperforms the discriminatory pricing mechanisms (both in terms of social welfare and the consumer's utility).

Hypothesis 3. When insufficient information is available and an inappropriate k is chosen, the uniform pricing mechanism can perform poorly and is consistently outperformed by the discriminatory pricing mechanisms.

Then, in Section 6.2.2, we turn to the Execution-Contingent VCG mechanism, which applies to cases where both the duration probabilities and costs of providers are private information. As this mechanism is efficient, i.e., always obtains the optimal solution, we are mostly concerned about the loss of utility caused by over-paying the providers. Since the mechanism uses more information about the providers, we test the following hypothesis:

Hypothesis 4. For a sufficiently large number of available providers, the Execution-Contingent VCG mechanism consistently matches or exceeds the consumer's utility achieved by the uniform and discriminatory pricing mechanisms.

Finally, as part of Section 6.2.2, we also consider the approximations introduced in Section 5.3.3 and investigate how they affect the consumer's performance.

6.2.1. Uniform and discriminatory pricing mechanisms

In order to test the performance of these mechanisms, we apply them to the same settings as described in Section 6.1.1.[25] Fig. 10 shows the results for two particular settings — one with many providers, a short deadline and a high value and one with fewer providers, a longer deadline and low value. These serve to illustrate the effect of varying the parameter k (before we move on to more general results over the whole range of environments). In more detail, for each strategy, Fig. 10 shows the utility obtained by the consumer and by the providers, as well as the efficiency loss, as compared to the optimal solution.

It is immediately obvious here that all mechanisms suffer from a loss in utility for the consumer, as described above. More specifically, if the optimal k is chosen for the uniform pricing mechanism, the consumer achieves an average 86.0% and 83.4% of the optimal (respectively in the two example settings). In terms of social welfare, this corresponds to 95.4% and 94.6% of the optimal. Compared to this, the discriminatory pricing mechanisms achieve an average consumer's utility that is 72.0% of the optimal, and an average social welfare that is 85.5% of the optimal. These results highlight that despite their simplicity, the proposed mechanisms can achieve a good performance compared to the optimal. Furthermore, choosing k optimally here clearly offers an advantage over the discriminatory mechanisms, thus supporting Hypothesis 2.

Our second observation here is that the performance of the uniform pricing mechanism depends heavily on the choice of k and can be as low as 25% of the optimal if the wrong parameter is chosen. Furthermore, the best parameter depends on

[25] As before, we solve the problem optimally when there are twelve or fewer providers and use the heuristic for settings with more providers.

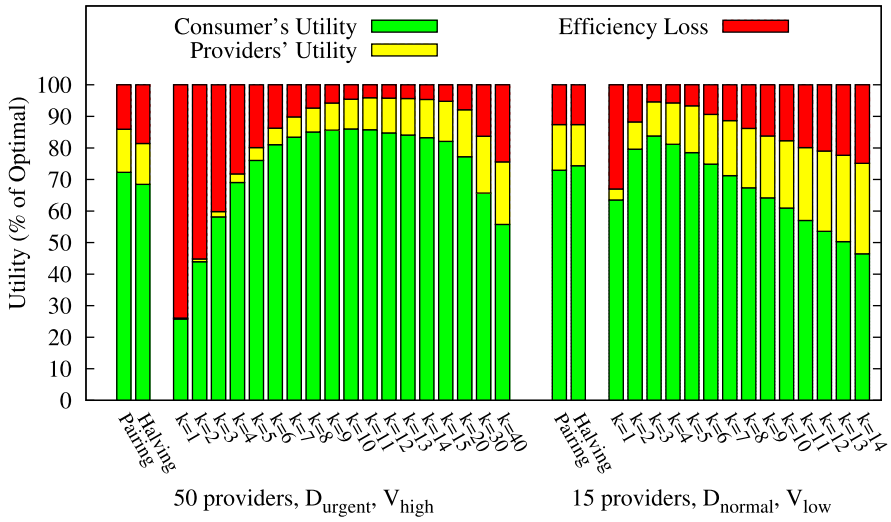

Fig. 10. Performance of incentive compatible mechanisms.

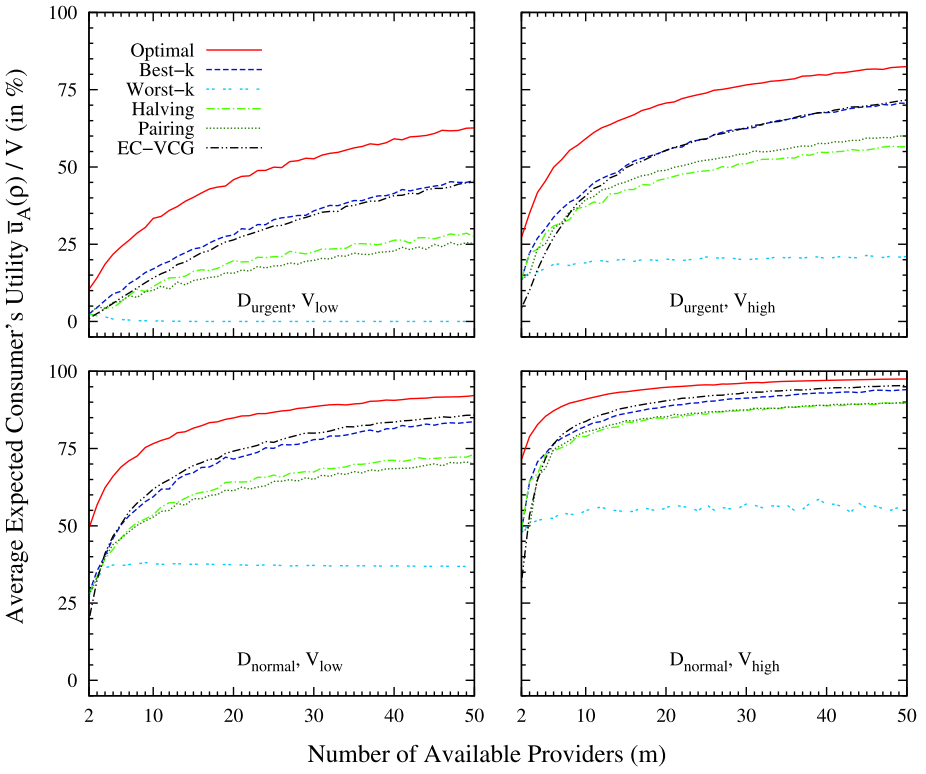

Fig. 11. Performance of incentive compatible mechanisms over all environments (average expected consumer's utility).

the scenario. For example, for the task with V_{low}, $k = 3$ is the best choice for maximising the consumer's utility, achieving over 83.4% of the optimal. However, for V_{high}, it is one of the worst, achieving only 58%. Hence, these results indicate that a consumer can achieve a good utility by using appropriate k parameters. However, when k is set incorrectly (e.g., when insufficient information is available about the environment), it can obtain better results by using one of the discriminatory pricing mechanisms, which consistently perform well. This supports Hypothesis 3.

Finally, to generalise these observations, Fig. 11 plots the performance of the mechanisms in terms of the average expected consumer's utility over all environments considered in this section.[26] In Fig. 12, we plot the corresponding average

[26] Note we discuss the performance of the Execution-Contingent VCG mechanism, labelled as EC-VCG, in Section 6.2.2.

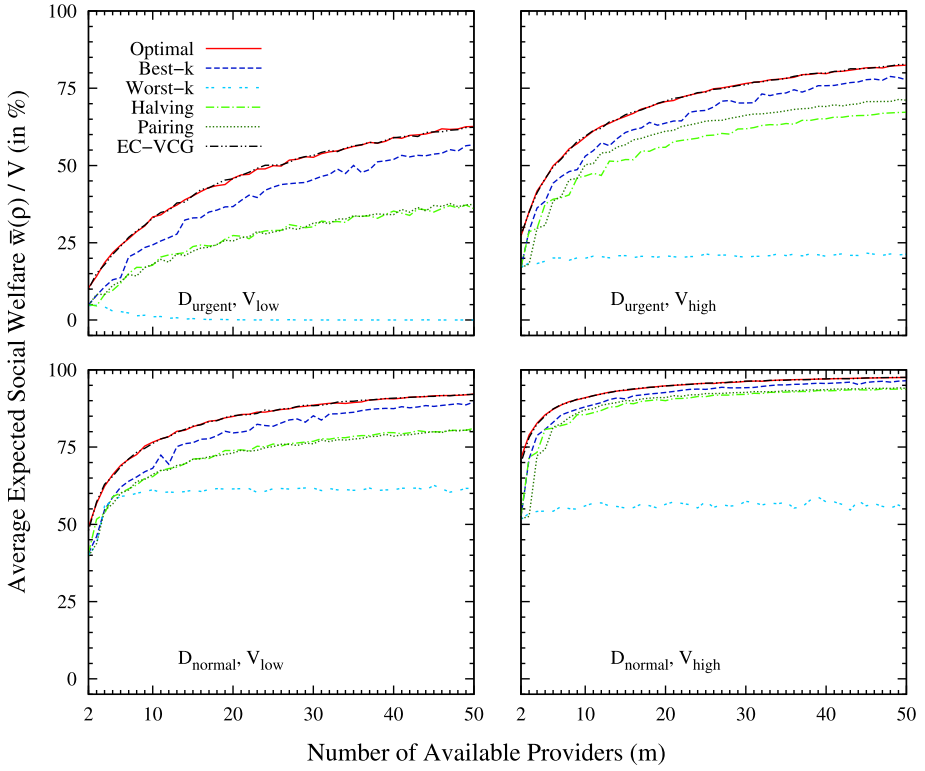

Fig. 12. Performance of incentive compatible mechanisms over all environments (average expected social welfare).

expected social welfare. For brevity, we replace all choices for k by a Best-k and a Worst-k mechanism, which show the performance when k is chosen to maximise and minimise the expected consumer's utility, respectively.[27] Here, we note that the discriminatory pricing mechanisms consistently perform worse than Best-k (supporting Hypothesis 2), but similarly outperform Worst-k (supporting Hypothesis 3). Neither of the two discriminatory pricing mechanisms dominates the other, but we generally observe that the Halving mechanism performs better in the low value (V_{low}) environments than the Pairing mechanism. This is due to the fact that the Halving mechanism usually results in at least one cheap provider at the expense of having fewer candidate providers to choose from. This is advantageous in the low value case, because the consumer prefers to invoke fewer and cheaper providers here. Furthermore, in Fig. 12, we notice that the mechanisms generally achieve a good overall social welfare.[28]

6.2.2. Execution-Contingent VCG

In this section, we now consider our final mechanism, the Execution-Contingent VCG, which can be applied in settings where both the duration distributions and costs are private information. Again, we apply this mechanism to the same settings as in the previous section.[29] The results of this are plotted in Figs. 11 (average expected consumer's utility) and 12 (average expected social welfare).

First, we note that since the Execution-Contingent VCG is efficient (i.e., always selects the social welfare maximising outcome), its average expected social welfare is equal to the performance of the optimal strategy in the full information setting. In terms of the consumer's utility, there are two prominent trends over all environments. Initially, when there are few providers, the Execution-Contingent VCG achieves a comparably low utility for the consumer. This is because each procured provider is more likely to make a significant contribution and thus receives a high payment from the mechanism (more specifically, the utility that would be obtained without that provider's presence is typically considerably lower).

[27] We implement this by considering the average results of all possible choices for k. We then pick the parameter k that yielded the highest or lowest average utility for the consumer and re-run the experiments with that parameter.

[28] The performance plots in that figure are not as smooth in the previous figures. This is because the mechanisms do not explicitly optimise for the expected social welfare, but rather for the consumer's utility. Thus, short downwards trends are sometimes observed, especially for the performance of the Best-k strategy, where a particular choice of k may be optimal for the consumer's utility, but not necessarily for the social welfare. This phenomenon is evidenced also by the first scenario in Fig. 10, where $k = 10$ maximises the consumer's utility, but $k = 11$ maximises the social welfare.

[29] Here, we again obtain results for thirteen or more providers using our heuristic approximation. In this particular setting, it must be noted that the mechanism is then no longer incentive compatible. However, as we argued earlier, the heuristic is a close approximation to the optimal case and so we nevertheless show the results to provide an *intuition* of the mechanism's performance.

Table 3
Summary of empirical results (compared to optimal social welfare in full information setting).

Mechanism	Consumer's utility (% of optimal)	Social welfare (% of optimal)
Uniform pricing (Best-k)	85.84%	94.37%
Discriminatory pricing (pairing)	76.70%	86.58%
Discriminatory pricing (halving)	77.34%	88.24%
Execution-Contingent VCG	85.59%	100.00%

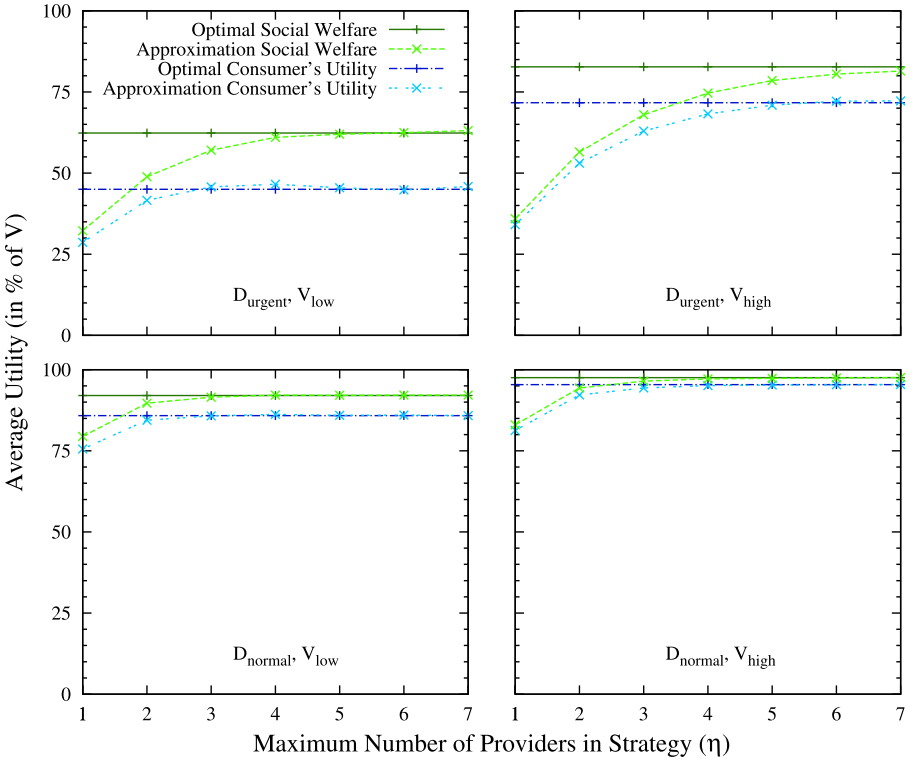

Fig. 13. Performance of Execution-Contingent VCG when approximating the optimal schedule.

However, as the number of providers in the system increases, the utility obtained by the Execution-Contingent VCG quickly rises, in most cases eventually exceeding the performance of all other mechanisms tested. In the first scenario (D_{urgent} and V_{low}), it only slowly approaches the performance of the Best-k mechanism, but as m is increased further, it also eventually dominates the latter (these results are not shown here). Overall, these results support Hypothesis 4. To summarise, the overall averaged performance results of all mechanisms shown in Figs. 11 and 12 are given in Table 3.

To conclude this section, we consider the effect of using approximations for the Execution-Contingent VCG mechanism. As argued in Section 5.3.3, such approximations allow the mechanism to remain incentive compatible even in settings with many agents. To this end, we consider the same environments examined thus far, but fix the number of total providers in the system to $m = 50$ (to consider a setting with many providers). We then record the performance of the Execution-Contingent VCG mechanism as we vary the maximum number of providers that are included in the allocation, i.e., the parameter η from Section 5.3.3. Both the average consumer's utility and the social welfare obtained are plotted in Fig. 13.

These results are positive, indicating that the mechanism performs well even if the number of providers in the final solution is restricted. In most scenarios, only two or three providers are required to perform close to the optimal, while the urgent and high value scenario requires four providers to achieve over 90% of the optimal social welfare. It is somewhat surprising here that the consumer's utility often reaches the equivalent utility of the optimal case more quickly and in some cases even exceeds it by a small amount. Intuitively, this is because using fewer providers increases the competition between those that are selected, thereby increasing the transfers they receive. This means that if the consumer is primarily interested in maximising its own profit rather than finding the efficient outcome, even fewer providers are required to achieve a good solution. Also, as the social welfare maximising outcome is not necessarily optimal for the consumer, it may even reap a small benefit by choosing η strategically. Furthermore, we note here that the time to find a solution is still fast even when choosing a high η. For example, when $\eta = 6$, it takes an average of six seconds in the urgent and high value

scenario; when $\eta = 7$, this rises to 45 seconds. These trends continue to hold for larger numbers of providers, and even problems with hundreds of providers can be solved in minutes or faster (see [16] for further results).

6.3. Choosing the right mechanism

To conclude this section, we now summarise our main theoretical and empirical results for each of the mechanisms. This builds on Table 2 and briefly describes the main information requirements, the scalability, expected performance (based on the results in this section) and other advantages and disadvantages of each mechanism. As such, it is intended to help a service consumer select a suitable mechanism for a particular setting.

- **Marginal contribution:**
  - *Information requirements*: Duration distributions need to be known by the consumer (e.g., through previous interactions or an appropriate trust and reputation system).
  - *Scalability*: Few providers (dozens or fewer), as it requires the consumer to compute an optimal procurement schedule.
  - *Performance*: Optimal social welfare, high utility to the consumer after transfers (around 85% of the optimal).
  - *Advantages*: Incentive compatible in dominant strategies.
  - *Disadvantages*: Vulnerable to de-commitments, which can be mitigated by imposing high penalties on non-compliant providers or by collecting a deposit prior to invocation. Also only individually rational in expectation, i.e., providers can sometimes incur a negative utility.
- **Uniform pricing:**
  - *Information requirements*: Known duration distributions and some knowledge about the providers' cost distributions. The latter is required to set the parameter k. More specifically, such knowledge would in practice arise from previous interactions with providers or general domain knowledge, and would allow the consumer to simulate the mechanism for various choices of k prior to execution.
  - *Scalability*: Many providers (hundreds or thousands).
  - *Performance*: Near-optimal social welfare (around 95% of the optimal), high consumer's utility (around 85% of the optimal).
  - *Advantages*: Incentive compatible in dominant strategies and post-execution individually rational (providers never make a loss).
  - *Disadvantages*: High information requirements.
- **Discriminatory pricing:**
  - *Information requirements*: Known duration distributions.
  - *Scalability*: Many providers (hundreds or thousands).
  - *Performance*: High social welfare (around 86–87% of the optimal), reasonable consumer's utility (around 77% of the optimal).
  - *Advantages*: Incentive compatible in dominant strategies and post-execution individually rational.
  - *Disadvantages*: Lower performance than other mechanisms.
- **Execution-Contingent VCG:**
  - *Information requirements*: None.
  - *Scalability*: Few providers (dozens).
  - *Performance*: Optimal social welfare, high consumer's utility (around 85% of the optimal).
  - *Advantages*: Low information requirements — elicits both truthful reports about costs and duration distributions.
  - *Disadvantages*: Ex-post incentive compatible (weaker than in dominant strategies) and only individually rational in expectation.
- **Approximate Execution-Contingent VCG:**
  - *Information Requirements*: None.
  - *Scalability*: Many providers (hundreds, depending on parameter η).
  - *Performance*: Close to Execution-Contingent VCG if sufficiently high η is chosen.
  - *Advantages*: Low information requirements.
  - *Disadvantages*: Ex-post incentive compatible and individually rational in expectation. Requires parameter η to be set by the consumer, but the choice is less critical than the parameter k of the uniform pricing mechanism. Specifically, it should generally be set to the highest feasible value given the computational constraints of the consumer. However, even with relatively low values (around $\eta = 6$ in the settings we tested), a performance that is almost equivalent to the optimal Execution-Contingent VCG can be achieved.

7. Conclusions

In this paper, we developed a novel approach for procuring services with uncertain execution durations in an optimal manner. This approach uses redundancy to increase the probability that a task is completed by a given deadline and explicitly balances the cost of redundancy with its benefit. In order to deal with large systems that may contain many (tens)

service providers, we proposed an optimal branch-and-bound algorithm to significantly reduce the search space and also a near-optimal greedy heuristic which can deal with hundreds or even thousands of providers.

We believe that our technique can be applied to a variety of realistic service-oriented settings, including grids and cloud computing, where services are offered by potentially unreliable providers. To highlight the benefit of using redundancy, we simulated these environments and showed empirically that it leads to a significant improvement in performance. In some cases, when the deadline of the task was short and the value high, using redundancy resulted in a 130% improvement over approaches that procure only a single provider, as is commonly done in existing task allocation scenarios. In our work, we investigated different sources of uncertainty and we showed that redundancy is most beneficial when service durations are independent. Surprisingly, however, redundancy can still offer a significant advantage even in settings where service durations are perfectly correlated (up to 27%).

Now, in order to procure services optimally, the consumer requires probabilistic performance information about the available services. While some existing work has considered the use of reputation systems in this context, such systems may not be available or there may be insufficient information about particular providers. To address such settings, we looked into techniques from mechanism design to incentivise providers to reveal their private performance information to the consumer. In particular, we developed a number of novel mechanisms that can be applied in different scenarios.

The first two of these, the uniform pricing mechanism and the discriminatory pricing mechanisms, apply to settings where only service costs are private information. These have highly desirable properties — they are incentive compatible in dominant strategies and are post-execution individually rational — and they retain these properties even when a sub-optimal outcome is chosen by the mechanism. For settings where costs and duration functions are private information, we showed that the standard VCG mechanism is no longer incentive compatible as providers can profit from inflating their response times. To address this, we proposed a novel Execution-Contingent VCG mechanism. This is ex-post incentive compatible, individually rational and efficient. However, unlike the other mechanisms, it has to select the optimal outcome, which can be intractable for very large environments. To address this, we developed a simple approximation, which can be found in polynomial time and which we demonstrated to achieve a high utility in practice.

We plan to extend our work in a number of ways in the future. First, we would like to consider settings with multiple interdependent tasks. These often occur in practical application areas where consumers need to complete complex workflows. To address these settings, we will extend our branch-and-bound algorithm and also consider more expressive consumer preferences, such as utility functions that depend on the workflow completion time.

Second, we will deal with multiple consumers that compete for the services of the providers, as this setting happens often in practical service-oriented application settings. For this, we plan to look into the application of two-sided markets and extend existing work in this area to our redundant procurement setting. In this context, we will be particularly interested in the impact increased competition among consumers will have on the use of redundancy. Clearly, when services are scarce, there will be less scope for redundant procurement, but particularly urgent or valuable tasks may still be allocated multiple services. Depending on the aims of the system designer, it will be interesting to also investigate alternative social welfare functions in this setting, such as the Nash product or egalitarian social welfare.

Finally, we intend to investigate more dynamic settings where both tasks and services arrive over time. This again is a common feature of realistic settings, where providers enter or leave the system and new unexpected tasks may suddenly appear. In such scenarios, agents may strategise not only about reporting their service capabilities, but also about when to make these reports. As such, we will consider applying and extending techniques from the field of online mechanism design [41].

Acknowledgements

We thank the reviewers of this paper for their detailed comments. This paper is a significantly extended version of previous conference publications [50] and [16]. The research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project, which is jointly funded by BAE Systems and EPSRC (EP/C548051/1), and as part of the EPSRC funded project on Market-Based Control (GR/T10664/01).

Appendix A. Optimal procurement as a restless bandit problem

The optimal procurement problem can be modelled as a restless bandit problem. Given c_i, $F_i(t)$ for each provider i, and deadline D, define state space $\mathcal{S} = \{S_{j,k} \mid j,k \geqslant 0, j+k \leqslant D\} \cup \{S_V, S_A\}$. For each provider i, define bandit B_i with starting state $S_{0,0}$. Define the activation probability function as follows. If the bandit is currently in state S_0, t with $t \leqslant D$ when activated, then with probability 1.0 it transitions to state $S_{t,1}$. If the current state is $S_{j,k}$ such that $j+k < D$, then it transitions to state $S_{j,k+1}$ with probability 1.0. Otherwise, if the bandit is in state $S_{j,k}$ such that $j+k = D$ then it transitions to state S_A, which is an absorbing state. The cost every time B_i is activated is c_i. The passive dynamics (i.e., when the bandit is not activated) are defined as follows. For any state $S_{0,k}$ such that $k \leqslant D$, the transition to state $S_{0,k+1}$ occurs with probability 1.0, and if $k = D$ then the transition to state S_A occurs with probability 1.0 (where S_A is again an absorbing state). For any state $S_{j,k}$ with $j > 0$ and $j+k \leqslant D$, the probability of transitioning to state S_V is $F_i(k)$ while the probability of transitioning to $S_{j,k+1}$ is $1 - F_i(k)$. The probability of transitioning from state S_V to state S_A and from any state $S_{j,k}$ such that $j+k = D$ to state S_A is also 1.0. Finally, set the passive rewards for all states except S_V to be zero, and set the

reward of S_V to be a *perishable* value V (i.e., once obtained, the value for the state changes to zero) [25]. This problem is defined such that a bandit will only be activated at most once in the optimal schedule, and the optimal activation schedule (i.e., that maximises the average reward) is exactly the optimal procurement strategy.

References

[1] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer, SETI@home: An experiment in public-resource computing, Communications of the ACM 45 (11) (November 2002) 56–61.
[2] M. Babaioff, M. Feldman, N. Nisan, Combinatorial agency, in: Proceedings of the 7th ACM Conference on Electronic Commerce (EC-2006), 2006, pp. 18–28.
[3] J.C. Beck, N. Wilson, Proactive algorithms for job shop scheduling with probabilistic durations, Journal of Artificial Intelligence Research 28 (2007) 183–232.
[4] D. Bergemann, S. Morris, Ex-post implementation, Games and Economic Behavior 63 (2) (2008) 527–566.
[5] R. Buyya, D. Abramson, S. Venugopal, The grid economy, Proceedings of the IEEE 93 (3) (2005) 698–714.
[6] P. Chalasani, S. Jha, O. Shehory, K. Sycara, Query restart strategies for web agents, in: Proceedings of the Second International Conference on Autonomous Agents (AGENTS '98), Minneapolis, USA, 1998, pp. 124–131.
[7] P. Cheeseman, B. Kanefsky, W.M. Taylor, Where the really hard problems are, in: Proceedings of the Twelfth International Joint Conference on (IJCAI-91), Sydney, Australia, 1991, pp. 331–337.
[8] D. Coit, A. Smith, Reliability optimization of series-parallel systems using a genetic algorithm, IEEE Transactions on Reliability 45 (2) (1996) 254–260.
[9] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.
[10] A. Erradi, P. Maheshwari, V. Tosic, Recovery policies for enhancing web services reliability, in: Proceedings of the IEEE Int. Conf. on Web Services (ICWS'06), Chicago, USA, 2006, pp. 189–196.
[11] L. Finkelstein, S. Markovitch, E. Rivlin, Optimal schedules for parallelizing anytime algorithms: The case of independent processes, in: Proceedings of the Eighteenth Conference on Artificial Intelligence (AAAI-02), Edmonton, Canada, 2002, pp. 719–724.
[12] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, International Journal of High Performance Computing Applications 15 (3) (August 2001) 200–222.
[13] T. Friese, J.P. Müller, B. Freisleben, Self-healing execution of business processes based on a peer-to-peer service architecture, in: Proceedings of the Eighteenth International Conference on Architecture of Computing Systems (ARCS '05), System Aspects in Organic and Pervasive Computing, Innsbruck, Austria, in: LNCS, vol. 3432, Springer, 2005, pp. 108–123.
[14] S. Garg, S. Venugopal, R. Buyya, A meta-scheduler with auction based resource allocation for global grids, in: Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS '08), Melbourne, Australia, 2008, pp. 187–194.
[15] E.H. Gerding, K. Larson, A. Rogers, N.R. Jennings, Mechanism design for task procurement with flexible quality of service, in: Proceedings of the 2009 International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2009), Budapest, Hungary, in: LNCS, vol. 5907, Springer, 2009, pp. 12–23.
[16] E.H. Gerding, S. Stein, K. Larson, A. Rogers, N.R. Jennings, Scalable mechanism design for the procurement of services with uncertain durations, in: Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '10), Toronto, Canada, 2010, pp. 649–656.
[17] J.C. Gittins, Bandit processes and dynamic allocation indices, Journal of the Royal Statistical Society. Series B (Methodological) 41 (2) (1979) 148–177.
[18] T. Glatard, J. Montagnat, X. Pennec, Optimizing jobs timeouts on clusters and production grids, in: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07), Rio de Janeiro, Brazil, 2007, pp. 100–107.
[19] C.P. Gomes, B. Selman, Algorithm portfolios, Artificial Intelligence 126 (2001) 43–62.
[20] C.-W. Hang, M. Singh, From quality to utility: Adaptive service selection framework, in: Proceedings of the 8th International Conference on Service Oriented Computing (ICSOC), San Francisco, USA, 2010.
[21] B. Hayes, Cloud computing, Communications of the ACM 51 (7) (2008) 9–11.
[22] B.A. Huberman, R.M. Lukose, T. Hogg, An economics approach to hard computational problems, Science 275 (5296) (1997) 51–54.
[23] M.N. Huhns, V.T. Holderfield, R.L.Z. Gutierrez, Robust software via agent-based redundancy, in: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '03), Melbourne, Australia, 2003, pp. 1018–1019.
[24] D.B. Ingham, F. Panzieri, S.K. Shrivastava, Advances in distributed systems, in: Advanced Distributed Computing: From Algorithms to Systems, in: LNCS, vol. 1752, Springer, Germany, 1999, pp. 277–294 (Ch. Constructing dependable Web services).
[25] P. Jacko, J. Niño-Mora, Time-constrained restless bandits and the knapsack problem for perishable items, Electronic Notes in Discrete Mathematics 28 (2007) 145–152.
[26] P. Jehiel, B. Moldovanu, Efficient design with interdependent valuations, Econometrica 69 (5) (2001) 1237–1259.
[27] H. Kautz, E. Horvit, Y. Ruan, C.P. Gomes, B. Selman, Dynamic restart policies, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 02), Edmonton, Canada, 2002, pp. 674–681.
[28] V. Krishna, Auction Theory, Academic Press, USA, 2002.
[29] R. Lavi, A. Mu'alem, N. Nisan, Two simplified proofs for Roberts' theorem, Social Choice and Welfare 32 (2009) 407–423.
[30] K. Leyton-Brown, E. Nudelman, Y. Shoham, Empirical hardness models: Methodology and a case study on combinatorial auctions, Journal of the ACM 56 (4) (2009) 1–52.
[31] L. Li, M.L. Littman, Lazy approximation for solving continuous finite-horizon MDPs, in: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05), Pittsburgh, USA, 2005, pp. 1175–1180.
[32] R.M. Lukose, B.A. Huberman, A methodology for managing risk in electronic transactions over the Internet, Netnomics 2 (1) (2000) 25–36.
[33] J. Marecki, M. Tambe, Towards faster planning with continuous resources in stochastic domains, in: Proceedings of the Twenty-Third Conference on Artificial Intelligence (AAAI-08), Chicago, USA, 2008, pp. 1049–1055.
[34] A. Mas-Colell, M. Whinston, J. Green, Microeconomic Theory, Oxford University Press, UK, 1995.
[35] Y. Narahari, R. Narayanam, D. Garg, H. Prakash, Mechanism design for resource procurement in grid computing, in: L. Jain, X. Wu (Eds.), Game Theoretic Problems in Network Economics and Mechanism Design Solutions. Advanced Information and Knowledge Processing, Springer, Germany, 2009, pp. 1–28.
[36] N. Nisan, A. Ronen, Computationally feasible VCG mechanisms, Journal of Artificial Intelligence Research 29 (2007) 19–47.
[37] Y. Oh, S.H. Son, Scheduling real-time tasks for dependability, The Journal of the Operational Research Society 48 (6) (1997) 629–639.
[38] T. Oinn, M. Greenwood, M. Addis, M.N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M.R. Pocock, M. Senger, R. Stevens, A. Wipat, C. Wroe, Taverna: lessons in creating a workflow environment for the life sciences, Concurrency and Computation: Practice and Experience 18 (10) (2005) 1067–1100.
[39] C.H. Papadimitriou, J.N. Tsitsiklis, The complexity of optimal queuing network control, Mathematics of Operations Research 24 (2) (1999) 293–305.
[40] A. Papakonstantinou, A. Rogers, E. Gerding, N. Jennings, Mechanism design for the truthful elicitation of costly probabilistic estimates in distributed information systems, Artificial Intelligence 175 (2) (February 2011) 648–672, http://eprints.ecs.soton.ac.uk/21316/.

[41] D.C. Parkes, Online mechanisms, in: N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani (Eds.), Algorithmic Game Theory, Cambridge University Press, UK, 2007, pp. 411–439 (Ch. 16).
[42] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, 2nd ed., Prentice Hall, USA, 2001.
[43] R. Porter, A. Ronen, Y. Shoham, M. Tennenholtz, Fault tolerant mechanism design, Artificial Intelligence 172 (15) (2008) 1783–1799.
[44] X. Qin, H. Jiang, D.R. Swanson, An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems, in: Proceedings of the International Conference on Parallel Processing (ICPP'02), Vancouver, Canada, 2002, pp. 360–368.
[45] S.D. Ramchurn, D. Huynh, N.R. Jennings, Trust in multiagent systems, Knowledge Engineering Review 19 (1) (2004) 1–25.
[46] S.D. Ramchurn, C. Mezzetti, A. Giovannucci, J.A. Rodriguez-Aguilar, R.K. Dash, N.R. Jennings, Trust-based mechanisms for robust and efficient task allocation in the presence of execution uncertainty, Journal of Artificial Intelligence Research 35 (2009) 119–159.
[47] D. Shahaf, E. Horvitz, Generalized task markets for human and machine computation, in: Proceedings of the Twenty-Fourth, AAAI Conference on Artificial Intelligence (AAAI-10), 2010, pp. 986–993.
[48] M.P. Singh, M.N. Huhns, Service-Oriented Computing: Semantics, Processes, Agents, John Wiley & Sons, Inc., USA, 2005.
[49] S. Stein, E.H. Gerding, N.R. Jennings, Optimal task migration in service-oriented systems: Algorithms and mechanisms, in: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10), Lisbon, Portugal, 2010, pp. 73–78.
[50] S. Stein, E.H. Gerding, A. Rogers, K. Larson, N.R. Jennings, Flexible procurement of services with uncertain durations using redundancy, in: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09), Pasadena, USA, 2009, pp. 292–298.
[51] S. Stein, N.R. Jennings, T.R. Payne, Flexible service provisioning with advance agreements, in: Proceedings of the Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08), Estoril, Portugal, 2008, pp. 249–256.
[52] S. Stein, T.R. Payne, N.R. Jennings, Flexible selection of heterogeneous and unreliable services in large-scale grids, Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences 367 (1897) (2009) 2483–2494.
[53] W.T.L. Teacy, J. Patel, N.R. Jennings, M. Luck, TRAVOS: Trust and reputation in the context of inaccurate information sources, Journal of Autonomous Agents and Multi-Agent Systems 12 (2) (February 2006) 183–198.
[54] F.A. Tillman, C.L. Hwang, W. Kuo, Optimization techniques for system reliability with redundancy: A review, IEEE Transactions on Reliability R-26 (1977) 148–155.
[55] K. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications, 2nd ed., John Wiley & Sons, Inc., USA, 2001.
[56] M.L. Weitzman, Optimal search for the best alternative, Econometrica 47 (3) (1979) 641–654.
[57] P. Whittle, Restless bandits, Journal of Applied Probability 25A (1988) 301–313.
[58] S.J. Witwicki, E.H. Durfee, Commitment-based service coordination, International Journal of Agent-Oriented Software Engineering 3 (1) (2009) 59–87.
[59] Y. Zhang, E. Manisterski, S. Kraus, V. Subrahmanian, D. Peleg, Computing the fault tolerance of multi-agent deployment, Artificial Intelligence 173 (3–4) (2009) 437–465.